Analysis of Algorithms
Outline

1 Algorithmic Complexity

2 Time Complexity

3 Space Complexity
Algorithmic Complexity

Time complexity of a program is how long it will take to solve a problem.

Space complexity of a program is how much memory it will need to solve a problem.
Algorithmic Complexity

Time complexity of a program is how long it will take to solve a problem
Algorithmic Complexity

Time complexity of a program is how long it will take to solve a problem.

Space complexity of a program is how much memory it will need to solve a problem.
Time Complexity

Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)
Program: ThreeSum.java
Program: ThreeSum.java

• Command-line input: a filename (String)
Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)
Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)
Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
$ cat ../data/1Kints.txt
```
Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
~/workspace/dsaj/programs
$ cat ../data/1Kints.txt
  324110
-442472
...
  745942
$ _
```
Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```
~/workspace/dsaj/programs

$ cat ../data/1Kints.txt
 324110
-442472
...
745942
$ /usr/bin/time -f "%s" java ThreeSum ../data/1Kints.txt
```
Time Complexity

Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
$ cat ../data/1Kints.txt
 324110
-442472
...
 745942
$ /usr/bin/time -f "%s" java ThreeSum ../data/1Kints.txt
70
0.28s
$ 
```
Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
$ cat ../data/1Kints.txt
  324110
 -442472
  ...
  745942
$ /usr/bin/time -f "%s" java ThreeSum ../data/1Kints.txt
  70
  0.28s
$ /usr/bin/time -f "%s" java ThreeSum ../data/2Kints.txt
```
Time Complexity

Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
> ~/workspace/dsaj/programs

$ cat ../data/1Kints.txt
 324110
-442472
...
 745942
$ /usr/bin/time -f "%s" java ThreeSum ../data/1Kints.txt
 70
 0.28s
$ /usr/bin/time -f "%s" java ThreeSum ../data/2Kints.txt
 528
 1.80s
$ _
Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
~/workspace/dsaj/programs

$ cat ../data/1Kints.txt
324110
-442472
...
745942
$ /usr/bin/time -f "%es" java ThreeSum ../data/1Kints.txt
70 0.28s
$ /usr/bin/time -f "%es" java ThreeSum ../data/2Kints.txt
528 1.80s
$ /usr/bin/time -f "%es" java ThreeSum ../data/4Kints.txt
```
Time Complexity

Program: ThreeSum.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```
$ ~/workspace/dsaj/programs

$ cat ../data/1Kints.txt
  324110
  -442472
  ...
  745942
$ /usr/bin/time -f "%s" java ThreeSum ../data/1Kints.txt
  70
  0.28s
$ /usr/bin/time -f "%s" java ThreeSum ../data/2Kints.txt
  528
  1.80s
$ /usr/bin/time -f "%s" java ThreeSum ../data/4Kints.txt
  4039
  14.06s
$ 
```
import stdlib.In;
import stdlib.StdOut;

public class ThreeSum {
    public static void main(String[] args) {
        In in = new In(args[0]);
        int[] a = in.readAllInts();
        StdOut.println(count(a));
    }

    private static int count(int[] a) {
        int n = a.length;
        int count = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                for (int k = j + 1; k < n; k++) {
                    if (a[i] + a[j] + a[k] == 0) {
                        count++;
                    }
                }
            }
        }
        return count;
    }
}
import stdlib.In;
import stdlib.StdOut;

public class ThreeSum {
    public static void main(String[] args) {
        In in = new In(args[0]);
        int[] a = in.readAllInts();
        StdOut.println(count(a));
    }

    private static int count(int[] a) {
        int n = a.length;
        int count = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                for (int k = j + 1; k < n; k++) {
                    if (a[i] + a[j] + a[k] == 0) {
                        count++;
                    }
                }
            }
        }
        return count;
    }
}
Experimental analysis

\[ f(n) = n^3 + 0.007625303 n^2 + 0.006868505 n + 0.01817256 \]
Experimental analysis

<table>
<thead>
<tr>
<th>$n$</th>
<th>$f(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>0.28s</td>
</tr>
<tr>
<td>2K</td>
<td>1.8s</td>
</tr>
<tr>
<td>4K</td>
<td>14.06s</td>
</tr>
<tr>
<td>8K</td>
<td>111.83s</td>
</tr>
<tr>
<td>16K</td>
<td>892.19s</td>
</tr>
</tbody>
</table>
Time Complexity

Experimental analysis

<table>
<thead>
<tr>
<th>n</th>
<th>f(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>0.28s</td>
</tr>
<tr>
<td>2K</td>
<td>1.8s</td>
</tr>
<tr>
<td>4K</td>
<td>14.06s</td>
</tr>
<tr>
<td>8K</td>
<td>111.83s</td>
</tr>
<tr>
<td>16K</td>
<td>892.19s</td>
</tr>
</tbody>
</table>

\[ f(n) = 0.2273121n^3 + 0.007625303n^2 + 0.006868505n + 0.01817256 \]
Time Complexity

The function $g(n)$ is called the tilde approximation of the function $f(n)$ if

$$\lim_{n \to \infty} g(n)f(n) = 1$$

For example, if $f(n) = 31n^2 + 78n + 42$, then $g(n) = 31n^2$. We often work with tilde approximations of the form $g(n) = an^b \log^c n$, where $a$, $b$, and $c$ are constants.

We refer to the function $T(n) = n^b \log^c n$ as the running time. For example, if $g(n) = 31n^2$, then $T(n) = n^2$.

For the Three Sum problem, $T(n) = n^3$. 


The function $g(n)$ is called the tilde approximation of the function $f(n)$ if

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 1$$

For example, if $f(n) = 31n^2 + 78n + 42$, then $g(n) = 31n^2$. We often work with tilde approximations of the form $g(n) = an^b \log^c n$, where $a$, $b$, and $c$ are constants. We refer to the function $T(n) = n^b \log^c n$ as the running time. For example, if $g(n) = 31n^2$, then $T(n) = n^2$. For the Three Sum problem, $T(n) = n^3$. 

**Time Complexity**
The function $g(n)$ is called the tilde approximation of the function $f(n)$ if

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 1$$

For example, if $f(n) = 31n^2 + 78n + 42$, then $g(n) = 31n^2$
The function $g(n)$ is called the tilde approximation of the function $f(n)$ if

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 1$$

For example, if $f(n) = 31n^2 + 78n + 42$, then $g(n) = 31n^2$

We often work with tilde approximations of the form $g(n) = an^b(\log n)^c$, where $a$, $b$, and $c$ are constants
**Time Complexity**

The function $g(n)$ is called the tilde approximation of the function $f(n)$ if

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 1$$

For example, if $f(n) = 31n^2 + 78n + 42$, then $g(n) = 31n^2$

We often work with tilde approximations of the form $g(n) = an^b(\log n)^c$, where $a$, $b$, and $c$ are constants

We refer to the function $T(n) = n^b(\log n)^c$ as the running time
Time Complexity

The function $g(n)$ is called the tilde approximation of the function $f(n)$ if

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = 1$$

For example, if $f(n) = 31n^2 + 78n + 42$, then $g(n) = 31n^2$

We often work with tilde approximations of the form $g(n) = an^b(\log n)^c$, where $a$, $b$, and $c$ are constants.

We refer to the function $T(n) = n^b(\log n)^c$ as the running time.

For example, if $g(n) = 31n^2$, then $T(n) = n^2$
Time Complexity

The function $g(n)$ is called the tilde approximation of the function $f(n)$ if

$$\lim_{{n \to \infty}} \frac{g(n)}{f(n)} = 1$$

For example, if $f(n) = 31n^2 + 78n + 42$, then $g(n) = 31n^2$

We often work with tilde approximations of the form $g(n) = an^b(\log n)^c$, where $a$, $b$, and $c$ are constants

We refer to the function $T(n) = n^b(\log n)^c$ as the running time

For example, if $g(n) = 31n^2$, then $T(n) = n^2$

For the Three Sum problem, $T(n) = n^3$
Time Complexity

Mathematically, we compute a function $f(n)$ from:

1. The cost of executing each statement (property of the computer)
2. The frequency of execution of each statement (property of the program and the input)
Mathematically, we compute a function $f(n)$ from:
Mathematically, we compute a function $f(n)$ from:

1. The cost of executing each statement (property of the computer)
Mathematically, we compute a function $f(n)$ from:

1. The cost of executing each statement (property of the computer)
2. The frequency of execution of each statement (property of the program and the input)
private static int count ( int [] a) {
    int n = a. length ;
    int count = 0;
    for ( int i = 0; i < n; i ++) {
        for ( int j = i + 1; j < n; j ++) {
            for ( int k = j + 1; k < n; k ++) {
                if (a[i] + a[j] + a[k] == 0) {
                    count ++;
                }
            }
        }
    }
    return count ;
}

Statement Block Time Frequency Total Time
A t 4 1 9
B t 3 n t 3 n
C t 2 (n^2) = n^2/2 - n/2
D t 1 (n^3) = n^3/6 - n^2/2 + n/3
E t 0 (depends on input)

f(n) = (t_1/6)n^3 + (t_2/2 - t_1/2)n^2 + (t_1/3 - t_2/2 + t_3)n + t_4 + t_0 x g(n) = (t_1/6)n^3
private static int count(int[] a) {
    int n = a.length;
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            for (int k = j + 1; k < n; k++) {
                if (a[i] + a[j] + a[k] == 0) {
                    count++;
                }
            }
        }
    }
    return count;
}
private static int count(int[] a) {
    int n = a.length;
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            for (int k = j + 1; k < n; k++) {
                if (a[i] + a[j] + a[k] == 0) {
                    count++;
                }
            }
        }
    }
    return count;
}

<table>
<thead>
<tr>
<th>Statement Block</th>
<th>Time</th>
<th>Frequency</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]</td>
<td>$t_4$</td>
<td>1</td>
<td>$t_4$</td>
</tr>
<tr>
<td>[B]</td>
<td>$t_3$</td>
<td>$n$</td>
<td>$t_3n$</td>
</tr>
<tr>
<td>[C]</td>
<td>$t_2$</td>
<td>$\binom{n}{2} = \frac{n^2}{2} - \frac{n}{2}$</td>
<td>$t_2(\frac{n^2}{2} - \frac{n}{2})$</td>
</tr>
<tr>
<td>[D]</td>
<td>$t_1$</td>
<td>$\binom{n}{3} = \frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3}$</td>
<td>$t_1(\frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3})$</td>
</tr>
<tr>
<td>[E]</td>
<td>$t_0$</td>
<td>$\times$ (depends on input)</td>
<td>$t_0\times$</td>
</tr>
</tbody>
</table>
```java
private static int count (int[] a) {
    int n = a.length; [A]
    int count = 0;
    for (int i = 0; i < n; i++) { [B]
        for (int j = i + 1; j < n; j++) { [C]
            for (int k = j + 1; k < n; k++) { [D]
                if (a[i] + a[j] + a[k] == 0) { [E]
                    count++;
                }
            }
        }
    }
    return count;
}
```

<table>
<thead>
<tr>
<th>Statement Block</th>
<th>Time</th>
<th>Frequency</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]</td>
<td>$t_4$</td>
<td>1</td>
<td>$t_4$</td>
</tr>
<tr>
<td>[B]</td>
<td>$t_3$</td>
<td>$n$</td>
<td>$t_3n$</td>
</tr>
<tr>
<td>[C]</td>
<td>$t_2$</td>
<td>$\binom{n}{2} = n^2/2 - n/2$</td>
<td>$t_2(n^2/2 - n/2)$</td>
</tr>
<tr>
<td>[D]</td>
<td>$t_1$</td>
<td>$\binom{n}{3} = n^3/6 - n^2/2 + n/3$</td>
<td>$t_1(n^3/6 - n^2/2 + n/3)$</td>
</tr>
<tr>
<td>[E]</td>
<td>$t_0$</td>
<td>$x$ (depends on input)</td>
<td>$t_0x$</td>
</tr>
</tbody>
</table>

$$f(n) = \left(\frac{t_1}{6}\right)n^3 + \left(\frac{t_2}{2} - \frac{t_1}{2}\right)n^2 + \left(\frac{t_1}{3} - \frac{t_2}{2} + t_3\right)n + t_4 + t_0x$$
private static int count(int[] a) {
    int n = a.length; // [A]
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) { // [B]
            for (int k = j + 1; k < n; k++) { // [C]
                if (a[i] + a[j] + a[k] == 0) { // [D]
                    count++; // [E]
                }
            }
        }
    }
    return count; // [F]
}

<table>
<thead>
<tr>
<th>Statement Block</th>
<th>Time</th>
<th>Frequency</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]</td>
<td>$t_4$</td>
<td>1</td>
<td>$t_4$</td>
</tr>
<tr>
<td>[B]</td>
<td>$t_3$</td>
<td>$n$</td>
<td>$t_3n$</td>
</tr>
<tr>
<td>[C]</td>
<td>$t_2$</td>
<td>$\binom{n}{2} = n^2/2 - n/2$</td>
<td>$t_2(n^2/2 - n/2)$</td>
</tr>
<tr>
<td>[D]</td>
<td>$t_1$</td>
<td>$\binom{n}{3} = n^3/6 - n^2/2 + n/3$</td>
<td>$t_1(n^3/6 - n^2/2 + n/3)$</td>
</tr>
<tr>
<td>[E]</td>
<td>$t_0$</td>
<td>$x$ (depends on input)</td>
<td>$t_0x$</td>
</tr>
</tbody>
</table>

$$f(n) = (t_1/6)n^3 + (t_2/2 - t_1/2)n^2 + (t_1/3 - t_2/2 + t_3)n + t_4 + t_0x$$

$$g(n) = (t_1/6)n^3$$
```java
private static int count(int[] a) {
    int n = a.length;  
    int count = 0;
    for (int i = 0; i < n; i++) {  
        for (int j = i + 1; j < n; j++) {  
            for (int k = j + 1; k < n; k++) {  
                if (a[i] + a[j] + a[k] == 0) {  
                    count++;  
                }
            }
        }
    }
    return count;
}
```

<table>
<thead>
<tr>
<th>Statement Block</th>
<th>Time</th>
<th>Frequency</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A]</td>
<td>$t_4$</td>
<td>1</td>
<td>$t_4$</td>
</tr>
<tr>
<td>[B]</td>
<td>$t_3$</td>
<td>$n$</td>
<td>$t_3n$</td>
</tr>
<tr>
<td>[C]</td>
<td>$t_2$</td>
<td>$\binom{n}{2} = \frac{n^2}{2} - \frac{n}{2}$</td>
<td>$t_2\left(\frac{n^2}{2} - \frac{n}{2}\right)$</td>
</tr>
<tr>
<td>[D]</td>
<td>$t_1$</td>
<td>$\binom{n}{3} = \frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3}$</td>
<td>$t_1\left(\frac{n^3}{6} - \frac{n^2}{2} + \frac{n}{3}\right)$</td>
</tr>
<tr>
<td>[E]</td>
<td>$t_0$</td>
<td>$x \cdot \text{(depends on input)}$</td>
<td>$t_0x$</td>
</tr>
</tbody>
</table>

$$f(n) = \left(\frac{t_1}{6}\right)n^3 + \left(\frac{t_2}{2} - \frac{t_1}{2}\right)n^2 + \left(\frac{t_1}{3} - \frac{t_2}{2} + t_3\right)n + t_4 + t_0x$$

$$g(n) = \left(\frac{t_1}{6}\right)n^3$$

$$T(n) = n^3$$
<table>
<thead>
<tr>
<th>Name</th>
<th>$T(n)$</th>
<th>Code Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>1</td>
<td>statement</td>
<td>increment the $i^{th}$ element in an array</td>
</tr>
<tr>
<td>logarithmic</td>
<td>$\log n$</td>
<td>divide and discard</td>
<td>binary search</td>
</tr>
<tr>
<td>linear</td>
<td>$n$</td>
<td>loop</td>
<td>find the maximum</td>
</tr>
<tr>
<td>linearithmic</td>
<td>$n \log n$</td>
<td>divide and conquer</td>
<td>merge sort</td>
</tr>
<tr>
<td>quadratic</td>
<td>$n^2$</td>
<td>double loop</td>
<td>check all ordered pairs</td>
</tr>
<tr>
<td>cubic</td>
<td>$n^3$</td>
<td>triple loop</td>
<td>check all ordered triples</td>
</tr>
<tr>
<td>exponential</td>
<td>$2^n$</td>
<td>exhaustive search</td>
<td>check all subsets</td>
</tr>
</tbody>
</table>
## Time Complexity

### Running time classifications

<table>
<thead>
<tr>
<th>Name</th>
<th>( T(n) )</th>
<th>Code Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant</td>
<td>1</td>
<td>statement</td>
<td>increment the ( i )th element in an array</td>
</tr>
<tr>
<td>logarithmic</td>
<td>( \log n )</td>
<td>divide and discard</td>
<td>binary search</td>
</tr>
<tr>
<td>linear</td>
<td>( n )</td>
<td>loop</td>
<td>find the maximum</td>
</tr>
<tr>
<td>linearithmic</td>
<td>( n \log n )</td>
<td>divide and conquer</td>
<td>merge sort</td>
</tr>
<tr>
<td>quadratic</td>
<td>( n^2 )</td>
<td>double loop</td>
<td>check all ordered pairs</td>
</tr>
<tr>
<td>cubic</td>
<td>( n^3 )</td>
<td>triple loop</td>
<td>check all ordered triples</td>
</tr>
<tr>
<td>exponential</td>
<td>( 2^n )</td>
<td>exhaustive search</td>
<td>check all subsets</td>
</tr>
</tbody>
</table>
Time Complexity

```
static int indexOf(Object[] a, Object key) returns the index of key in the array a, or -1
static int indexOf(int[] a, int key) returns the index of key in the array a, or -1
static int indexOf(double[] a, double key) returns the index of key in the array a, or -1
```
### Time Complexity

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>static int indexOf(Object[] a, Object key)</code></td>
<td>returns the index of <code>key</code> in the array <code>a</code>, or -1</td>
</tr>
<tr>
<td><code>static int indexOf(int[] a, int key)</code></td>
<td>returns the index of <code>key</code> in the array <code>a</code>, or -1</td>
</tr>
<tr>
<td><code>static int indexOf(double[] a, double key)</code></td>
<td>returns the index of <code>key</code> in the array <code>a</code>, or -1</td>
</tr>
</tbody>
</table>
Time Complexity
Program: LinearSearch.java
Program: LinearSearch.java

- Command-line input: a filename (String)
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```bash
> ~/workspace/dsaj/programs

$ _
```
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```
> ~/workspace/dsaj/programs

$ cat ../data/tinyW.txt
```
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```bash
$ cat../data/tinyW.txt
84
48
...
29
$ cat../data/tinyT.txt
```
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

$ ~/workspace/dsaj/programs

$ cat ../data/tinyW.txt
84
48
...
29
$ cat ../data/tinyT.txt
23
50
...
68
$ _
Program: LinearSearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```
> ~workspace/dsaj/programs

$ cat ../data/tinyW.txt
84
48
...
29
$ cat ../data/tinyT.txt
23
50
...
68
$ java dsa.LinearSearch ../data/tinyW.txt < ../data/tinyT.txt
```
**Time Complexity**

**Program:** `LinearSearch.java`

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```
$ ~/workspace/dsaj/programs
$ cat ../data/tinyW.txt
84
48
...
29
$ cat ../data/tinyT.txt
23
50
...
68
$ java dsa.LinearSearch ../data/tinyW.txt < ../data/tinyT.txt
50
99
13
$ _
```
package dsa;
import stdlib.In;
import stdlib.StdIn;
import stdlib.StdOut;

public class LinearSearch {

    public static int indexOf(Object[] a, Object key) {
        for (int i = 0; i < a.length; i++) {
            if (a[i].equals(key)) {
                return i;
            }
        }
        return -1;
    }

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

    public static int indexOf(double[] a, double key) {
        for (int i = 0; i < a.length; i++) {
            if (a[i] == key) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {

    }
}
package dsa;

import stdlib.In;
import stdlib.StdIn;
import stdlib.StdOut;

public class LinearSearch {
    public static int indexOf(Object[] a, Object key) {
        for (int i = 0; i < a.length; i++) {
            if (a[i].equals(key)) {
                return i;
            }
        }
        return -1;
    }

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

    public static int indexOf(double[] a, double key) {
        for (int i = 0; i < a.length; i++) {
            if (a[i] == key) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {

    }
In inStream = new In(args[0]);
int[] whiteList = inStream.readAllInts();
while (!StdIn.isEmpty()) {
    int key = StdIn.readInt();
    if (indexOf(whiteList, key) == -1) {
        StdOut.println(key);
    }
}

}
Time Complexity

ds.BinarySearch

- static int indexOf(Comparable[] a, Comparable key) returns the index of key in the sorted array a, or -1
- static int indexOf(int[] a, int key) returns the index of key in the sorted array a, or -1
- static int indexOf(double[] a, double key) returns the index of key in the sorted array a, or -1
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static int indexOf(Comparable[] a, Comparable key)</td>
<td>returns the index of key in the sorted array a, or -1</td>
</tr>
<tr>
<td>static int indexOf(int[] a, int key)</td>
<td>returns the index of key in the sorted array a, or -1</td>
</tr>
<tr>
<td>static int indexOf(double[] a, double key)</td>
<td>returns the index of key in the sorted array a, or -1</td>
</tr>
</tbody>
</table>
Time Complexity
Program: BinarySearch.java
Program: BinarySearch.java

- Command-line input: a filename (String)
Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file
Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file
Time Complexity

Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```
> ~/workspace/dsaj/programs

$ cat ../data/tinyW.txt
```
Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

> ~/workspace/dsaj/programs

$ cat ../data/tinyW.txt
84
48
...
29
$ _
Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```bash
> ~/workspace/dsaj/programs

$ cat ../data/tinyW.txt
84
48
...
29
$ cat ../data/tinyT.txt
```
Time Complexity

Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

> ~/workspace/dsaj/programs

$ cat ../data/tinyW.txt
84
48
...
29
$ cat ../data/tinyT.txt
23
50
...
68
$ _
Time Complexity

Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```bash
$ cat ../data/tinyW.txt
84
48
...
29
$ cat ../data/tinyT.txt
23
50
...
68
$ java dsa.BinarySearch ../data/tinyW.txt < ../data/tinyT.txt
```
Program: BinarySearch.java

- Command-line input: a filename (String)
- Standard input: a sequence of integers
- Standard output: the integers from standard input that are not in the file

```bash
> ~/workspace/dsaj/programs
$ cat ../data/tinyW.txt
84
48
...
29
$ cat ../data/tinyT.txt
23
50
...
68
$ java dsa.BinarySearch ../data/tinyW.txt < ../data/tinyT.txt
50
99
13
$ _
```
Time Complexity

Successful binary search for the key 23 (returns 5)
Successful binary search for the key 23 (returns 5)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>a[]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>23</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>29</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>33</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>48</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>54</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>57</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>68</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>77</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>84</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>98</td>
<td>84</td>
</tr>
</tbody>
</table>
**Time Complexity**

Successful binary search for the key 23 (returns 5)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>14</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
</tr>
</tbody>
</table>
Successful binary search for the key 23 (returns 5)
Successful binary search for the key 23 (returns 5)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>a[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>23</td>
<td>29</td>
</tr>
<tr>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
</tr>
<tr>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
</tr>
</tbody>
</table>
Successful binary search for the key 23 (returns 5)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
</tr>
</tbody>
</table>
Time Complexity

Unsuccessful binary search for the key 50 (returns -1)
Time Complexity

Unsuccessful binary search for the key 50 (returns -1)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>a[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>48</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>54</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>57</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>68</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>77</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>98</td>
</tr>
</tbody>
</table>
Unsuccessful binary search for the key 50 (returns -1)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
<td>14</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
</tr>
</tbody>
</table>
Unsuccessful binary search for the key 50 (returns -1)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>11</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>a[]</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
</tr>
</tbody>
</table>
### Time Complexity

Unsuccessful binary search for the key 50 (returns -1)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
</tr>
</tbody>
</table>
Time Complexity

Unsuccessful binary search for the key 50 (returns -1)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
</tr>
</tbody>
</table>
Unsuccessful binary search for the key 50 (returns -1)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
<td></td>
</tr>
</tbody>
</table>
Unsuccessful binary search for the key 50 (returns -1)

<table>
<thead>
<tr>
<th>lo</th>
<th>mid</th>
<th>hi</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>16</td>
<td>18</td>
<td>23</td>
<td>29</td>
<td>33</td>
<td>48</td>
<td>54</td>
<td>57</td>
<td>68</td>
<td>77</td>
<td>84</td>
<td>98</td>
<td></td>
</tr>
</tbody>
</table>
package dsa;

import java.util.Arrays;
import stdlib.In;
import stdlib.StdIn;
import stdlib.StdOut;

public class BinarySearch {

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

    public static int indexOf(int[] a, int key) {
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) {
            int mid = lo + (hi - lo) / 2;
            if (key < a[mid]) {
                hi = mid - 1;
            } else if (key > a[mid]) {
                lo = mid + 1;
            } else {
                return mid;
            }
        }
        return -1;
    }
}
public class BinarySearch {
    public static int indexOf(Comparable[] a, Comparable key) {
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) {
            int mid = lo + (hi - lo) / 2;
            int cmp = key.compareTo(a[mid]);
            if (cmp < 0) {
                hi = mid - 1;
            } else if (cmp > 0) {
                lo = mid + 1;
            } else {
                return mid;
            }
        }
        return -1;
    }

    public static int indexOf(int[] a, int key) {
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) {
            int mid = lo + (hi - lo) / 2;
            if (key < a[mid]) {
                hi = mid - 1;
            } else if (key > a[mid]) {
                lo = mid + 1;
            } else {
                return mid;
            }
        }
        return -1;
    }
}
```java
public static int indexOf(double[] a, double key) {
    int lo = 0;
    int hi = a.length - 1;
    while (lo <= hi) {
        int mid = lo + (hi - lo) / 2;
        if (key < a[mid]) {
            hi = mid - 1;
        } else if (key > a[mid]) {
            lo = mid + 1;
        } else {
            return mid;
        }
    }
    return -1;
}

public static void main(String[] args) {
    In inStream = new In(args[0]);
    int[] whiteList = inStream.readAllInts();
    Arrays.sort(whiteList);
    while (!StdIn.isEmpty()) {
        Integer key = StdIn.readInt();
        if (indexOf(whiteList, key) == -1) {
            StdOut.println(key);
        }
    }
}
```
Time Complexity

The running time of a single linear search on an array of size $n$ is $T(n) = n$.

The running time of a single binary search on an array of size $n$ is $T(n) = n \log n$ (sorting cost) + $\log n$ (searching cost).

The running time of $m$ linear searches on an array of size $n$ is $T(n) = mn$.

The running time of $m$ binary searches on an array of size $n$ is $T(n) = n \log n$ (sorting cost) + $m \log n$ (searching cost).
Time Complexity

The running time of a single linear search on an array of size $n$ is

$$T(n) = n$$
Time Complexity

The running time of a single linear search on an array of size $n$ is

$$T(n) = n$$

The running time of a single binary search on an array of size $n$ is

$$T(n) = n \log n \text{ (sorting cost)} + \log n \text{ (searching cost)}$$
The running time of a single linear search on an array of size $n$ is

$$T(n) = n$$

The running time of a single binary search on an array of size $n$ is

$$T(n) = n \log n \text{ (sorting cost)} + \log n \text{ (searching cost)}$$

The running time of $m$ linear searches on an array of size $n$ is

$$T(n) = mn$$
**Time Complexity**

The running time of a single linear search on an array of size \( n \) is

\[
T(n) = n
\]

The running time of a single binary search on an array of size \( n \) is

\[
T(n) = n \log n \text{ (sorting cost)} + \log n \text{ (searching cost)}
\]

The running time of \( m \) linear searches on an array of size \( n \) is

\[
T(n) = mn
\]

The running time of \( m \) binary searches on an array of size \( n \) is

\[
T(n) = n \log n \text{ (sorting cost)} + m \log n \text{ (searching cost)}
\]
Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)
Program: ThreeSumFast.java
Program: ThreeSumFast.java

- Command-line input: a filename (String)
Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)
Time Complexity

Program: ThreeSumFast.java

• Command-line input: a filename (String)
• Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```
> ~/workspace/dsaj/programs

$ _
```
Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```
~/workspace/dsaj/programs
$ /usr/bin/time -f "%s" java ThreeSumFast ../data/1Kints.txt
```
Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
~/workspace/dsaj/programs
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/1Kints.txt
70
0.10s
$ _
```
Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```
> ~/workspace/dsaj/programs

$ /usr/bin/time -f "%es" java ThreeSumFast ../data/1Kints.txt
70
0.10s
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/2Kints.txt
```
Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```
> ~/workspace/dsaj/programs
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/1Kints.txt
70
0.10s
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/2Kints.txt
528
0.17s
$ 
```
Time Complexity

Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```
> ~/workspace/dsaj/programs
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/1Kints.txt
70
0.10s
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/2Kints.txt
528
0.17s
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/4Kints.txt
```
Program: ThreeSumFast.java

- Command-line input: a filename (String)
- Standard output: the number of unordered triples \((x, y, z)\) in the file such that \(x + y + z = 0\)

```bash
> ~/workspace/dsaj/programs
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/1Kints.txt
70
0.10s
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/2Kints.txt
528
0.17s
$ /usr/bin/time -f "%es" java ThreeSumFast ../data/4Kints.txt
4039
0.47s
$ _
```
import java.util.Arrays;
import dsa.BinarySearch;
import stdlib.In;
import stdlib.StdOut;

public class ThreeSumFast {
    public static void main(String[] args) {
        In in = new In(args[0]);
        int[] a = in.readAllInts();
        StdOut.println(count(a));
    }

    private static int count(int[] a) {
        int n = a.length;
        Arrays.sort(a);
        int count = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                int k = BinarySearch.indexOf(a, -(a[i] + a[j]));
                if (k > j) {
                    count++;
                }
            }
        }
        return count;
    }
}

Time Complexity
public class ThreeSumFast {
    public static void main(String[] args) {
        In in = new In(args[0]);
        int[] a = in.readAllInts();
        StdOut.println(count(a));
    }

    private static int count(int[] a) {
        int n = a.length;
        Arrays.sort(a);
        int count = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                for (int k = 1; k < n; k++) {
                    int k = BinarySearch.indexOf(a, -(a[i] + a[j]));
                    if (k > j) {
                        count++;
                    }
                }
            }
        }
        return count;
    }
}
<table>
<thead>
<tr>
<th>Size</th>
<th>Fast Three Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>0.28s</td>
</tr>
<tr>
<td>2K</td>
<td>1.8s</td>
</tr>
<tr>
<td>4K</td>
<td>14.06s</td>
</tr>
<tr>
<td>8K</td>
<td>111.83s</td>
</tr>
<tr>
<td>16K</td>
<td>892.19s</td>
</tr>
</tbody>
</table>
## Time Complexity

<table>
<thead>
<tr>
<th>$n$</th>
<th>Three Sum $T(n)$</th>
<th>Fast Three Sum $T(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1K</td>
<td>0.28s</td>
<td>0.1s</td>
</tr>
<tr>
<td>2K</td>
<td>1.8s</td>
<td>0.17s</td>
</tr>
<tr>
<td>4K</td>
<td>14.06s</td>
<td>0.47s</td>
</tr>
<tr>
<td>8K</td>
<td>111.83s</td>
<td>1.58s</td>
</tr>
<tr>
<td>16K</td>
<td>892.19s</td>
<td>6.09s</td>
</tr>
</tbody>
</table>
## Space Complexity

**Memory requirements for primitive types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>1</td>
</tr>
<tr>
<td>byte</td>
<td>1</td>
</tr>
<tr>
<td>char</td>
<td>2</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
</tr>
<tr>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>float</td>
<td>4</td>
</tr>
<tr>
<td>long</td>
<td>8</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
</tr>
</tbody>
</table>

To determine the memory usage of an object, we add the amount of memory used by each instance variable. For example, a Counter object uses 12 bytes: 8 bytes for `id` (a reference) and 4 bytes for `count`.
Space Complexity

Memory requirements for primitive types

<table>
<thead>
<tr>
<th>Type</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>1</td>
</tr>
<tr>
<td>byte</td>
<td>1</td>
</tr>
<tr>
<td>char</td>
<td>2</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
</tr>
<tr>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>float</td>
<td>4</td>
</tr>
<tr>
<td>long</td>
<td>8</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
</tr>
</tbody>
</table>
Memory requirements for primitive types

<table>
<thead>
<tr>
<th>Type</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>1</td>
</tr>
<tr>
<td>byte</td>
<td>1</td>
</tr>
<tr>
<td>char</td>
<td>2</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
</tr>
<tr>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>float</td>
<td>4</td>
</tr>
<tr>
<td>long</td>
<td>8</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
</tr>
</tbody>
</table>

To determine the memory usage of an object, we add the amount of memory used by each instance variable.
Space Complexity

Memory requirements for primitive types

<table>
<thead>
<tr>
<th>Type</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>1</td>
</tr>
<tr>
<td>byte</td>
<td>1</td>
</tr>
<tr>
<td>char</td>
<td>2</td>
</tr>
<tr>
<td>short</td>
<td>2</td>
</tr>
<tr>
<td>int</td>
<td>4</td>
</tr>
<tr>
<td>float</td>
<td>4</td>
</tr>
<tr>
<td>long</td>
<td>8</td>
</tr>
<tr>
<td>double</td>
<td>8</td>
</tr>
</tbody>
</table>

To determine the memory usage of an object, we add the amount of memory used by each instance variable.

For example, a `Counter` object uses 12 bytes: 8 bytes for `id` (a reference) and 4 bytes for `count`.
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values. For example, an array of \( n \) \textit{int} values uses 4\( n \) bytes. An array of objects is an array of references to objects, so we need to add the space for the references to the space required for the objects. For example, an array of \( n \) \textit{Counter} objects uses 8\( n \) bytes for references plus 12 bytes for each \textit{Counter} object, for a grand total of 20\( n \) bytes.

A 2D array is an array of arrays (each array is an object). For example, an \( m \times n \) array of \textit{double} values uses 8\( m \) bytes for references plus 8 bytes for each of the \( m n \) \textit{double} values, for a grand total of 8\( mn \) + 8\( m \) bytes.

A \textit{String} of length \( n \) uses 2\( n \) bytes.
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values.
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values.

For example, an array of $n \text{ int}$ values uses $4n$ bytes.
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values.

For example, an array of \( n \) \texttt{int} values uses \( 4n \) bytes.

An array of objects is an array of references to objects, so we need to add the space for the references to the space required for the objects.
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values.

For example, an array of \( n \) int values uses \( 4n \) bytes.

An array of objects is an array of references to objects, so we need to add the space for the references to the space required for the objects.

For example, an array of \( n \) Counter objects uses \( 8n \) bytes for references plus 12 bytes for each Counter object, for a grand total of \( 20n \) bytes.
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values.

For example, an array of $n$ int values uses $4n$ bytes.

An array of objects is an array of references to objects, so we need to add the space for the references to the space required for the objects.

For example, an array of $n$ Counter objects uses $8n$ bytes for references plus 12 bytes for each Counter object, for a grand total of $20n$ bytes.

A 2D array is an array of arrays (each array is an object).
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values.

For example, an array of \( n \) int values uses \( 4n \) bytes.

An array of objects is an array of references to objects, so we need to add the space for the references to the space required for the objects.

For example, an array of \( n \) Counter objects uses \( 8n \) bytes for references plus 12 bytes for each Counter object, for a grand total of \( 20n \) bytes.

A 2D array is an array of arrays (each array is an object).

For example, an \( m \times n \) array of double values uses \( 8m \) bytes for references plus 8 bytes for each of the \( mn \) double values, for a grand total of \( 8mn + 8m \) bytes.
Space Complexity

The memory requirement for an array of primitive-type values is the memory needed to store the values.

For example, an array of $n$ int values uses $4n$ bytes.

An array of objects is an array of references to objects, so we need to add the space for the references to the space required for the objects.

For example, an array of $n$ Counter objects uses $8n$ bytes for references plus 12 bytes for each Counter object, for a grand total of $20n$ bytes.

A 2D array is an array of arrays (each array is an object).

For example, an $m \times n$ array of double values uses $8m$ bytes for references plus 8 bytes for each of the $mn$ double values, for a grand total of $8mn + 8m$ bytes.

A String of length $n$ uses $2n$ bytes.