Command Line Arguments

public static void main (String[] args)

allows the user to provide input to the program without interacting (via console) with the program

To set

On eclipse: Run ⇒ Run Configurations ⇒ Arguments Tab ⇒ Program Arguments

infile.txt  outfile.txt  1000

args [0]: infile.txt  each stored
args [1]: outfile.txt  as String
args [2]: 1000

To convert integer (double) to String:

int numInters = Integer.parseInt(args[2]);
Running Time Analysis

Program, input size = n (ex: n numbers to sort).

Goals:
1) Express runtime of a program as a function of input size n:
   \[ T(n) \text{ runtime of a program in terms of } n \]
   \[ f(n) \text{ some function of } n \text{ (ex: } n^2, n^3, \log n) \]

2) Establish relative order between \( T(n) \) and \( f(n) \).

How to compare 2 functions?
Can't just say one is greater than the other.
ex: \( f(n) = 1000n \), \( g(n) = n^2 \)

\[ f(n) > g(n) \text{ for } n < 1000 \]
\[ f(n) < g(n) \text{ for } n > 1000 \]

So instead, compare in terms of relative growth rates

growth rate - how fast a function grows asymptotically (i.e. as \( n \to \infty \)).

[Diagram]

fastest growing.
Order of typical growth rates

- constant
- logarithmic
- log-squared
- linear
- $n \log n$
- quadratic
- cubic
- polynomial
- exponential

```
"Big-on"
```

1. $T(n) = \Theta(f(n))$ - $T(n)$ grows at a rate $\leq c f(n)$ for $c > 0$
2. $T(n)$ grows no faster than $f(n)$.

ex: $T(n) = 1 + 5n$, $T(n) = 1 + 5n^2$

First, which grows faster? $T(n) = 1 + 5n^2$
So this corresponds to the slower program.

**How to find $f(n)$**

1. Pick out fastest growing term in $T(n)$
2. Drop coefficients

ex: $T(n) = 1 + 5n$; $T(n) = 1 + 5n^2$

- $f(n) = n$
- $f(n) = n^2$

.: $T(n) = \Theta(n)$
.: $T(n) = \Theta(n^2)$
3. \( T(n) = 5 + 20n + 3n^2 \)
\[ T(n) = O(n^2) \]

(2) and (3) grow at same rate.

4. \( T(n) = \frac{n^2(n^2+1)}{2} = \frac{n^4 + n^2}{2} \)
\[ f(n) = n^4 \]
\[ T(n) = O(n^4) \]

5. \( T(n) = 5 \)
\[ f(n) = 1 \]
\[ T(n) = O(1) \]

6. \( T(n) = n(5 + \log n) \)
\[ = 5n + n\log n \]
\[ f(n) = n\log n \]
\[ T(n) = O(n\log n) \]
In these examples, we have a formula for $T(n)$. But how to find $T(n)$ of a program?

General rules

1. For loop:
   
   for (n iterations)
   
   body
   
   $T(n) = T(body) \times n$

2. Nested for loop:
   
   for (n iterations)
   
   for (m iterations)
   
   body
   
   $T(n) = T(body) \times n \times m$
3. Consecutive Statements

statement 1 \{ a statement may be a for loop, while loop, print, call to a method, etc.
statement 2
statement k

T(n) = \max (T(\text{statement 1}), T(\text{statement 2}), \ldots, T(\text{statement k}))

ex. \( S_1 = n^2 \), \( S_2 = n \), \( S_3 = 2^n \)

\[ T(n) = 2^n \]

4. If/else

if (condition)
\<\text{body 1}\>
else
\<\text{body 2}\>

\[ T(n) = \max (T(\text{condition}), T(\text{body 1}), T(\text{body 2})) \]
Exs.

1. for (i = 0 \rightarrow n)
   for (j = 0 \rightarrow m)
     \text{sum} \leftarrow \text{sum} + 1

\[ T(n) = 1 + nm \]
\[ = O(nm) \]

2. for (i = 0 \rightarrow n)
   for (j = 0 \rightarrow n)
     for (k = 0 \rightarrow m)
       \text{sum} \leftarrow \text{sum} + 1
     \text{prod} \leftarrow \text{prod} \cdot \text{sum}

\[ T(n) = 2 + n^2 m \]
\[ = O(n^2 m) \]

3. for (i = 0, i \leq n, i++)
   for (j = 0, j \leq i, j++)
     \text{sum} \leftarrow \text{sum} + 1

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th># sum+</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>n-1</td>
<td>0</td>
<td>n</td>
</tr>
</tbody>
</table>

\[ T(n) = \text{total # sum} + 1 + 2 + 3 + \cdots + n \]
\[ = \frac{n}{2} \cdot \frac{i}{i} = \frac{n(n+1)}{2} - \frac{O(n^2)}{2} \]
(6) A: array of size n

if (i <= A[0] && i <= A[1] && ... && i <= A[n-1])
    sum++;
else
    sum--;

T(n) = max(n + 1 + 1)
     = O(n)

(7) if ...

    for (i = 0 to n)
        sum++;

T(n) = max(n + n + 1)
     = O(n)

(9) i = n
    while (i > 1)
        i = i / 2