Analysis of Algorithms

function ContainsDuplicate

Input : Array A of n elements.

```
function ContainsDuplicate(A[ ],$n$)
for $i \leftarrow 2$ to $n$ do
    for $j \leftarrow 1$ to $i - 1$ do
        if $(A[i] = A[j])$ then
            return (true);
        end
    end
end
return (false);
```

function ContainsDuplicate **(II)**

Input : Array A of n elements.

```
function ContainsDuplicate(A[ ],$n$)
for $i \leftarrow 2$ to $n$ do
    for $j \leftarrow 1$ to $i - 1$ do
        if $(A[i] = A[j])$ then
            return (true);
        end
    end
end
return (false);
```

Sample for loops

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $\lfloor \sqrt{i} \rfloor$ do
        $x \leftarrow x + (i - j)$;
    end
end
return (x);
```

Sample for loops (II)

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $\lfloor\sqrt{i}\rfloor$ do
        $x \leftarrow x + (i - j)$;
    end
end
return (x);
```

Sample for loops

```
**function** `func`($n$)
$x \leftarrow 0$;
**for** $i \leftarrow 1$ **to** $\lfloor\sqrt{n}\rfloor$ **do**
    **for** $j \leftarrow 1$ **to** $\lfloor\sqrt{n}\rfloor - i$ **do**
        $x \leftarrow x + (i - j)$;
    **end**
**end**
**return** (x);
```

Sample for loops (II)

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $\lfloor\sqrt{n}\rfloor$ do
    for $j \leftarrow 1$ to $\lfloor\sqrt{n}\rfloor - i$ do
        $x \leftarrow x + (i - j)$;
    end
end
return (x);
```

Sample while loops

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 7$;
while ($i \leq n$) do
    $x \leftarrow x + i$;
    $i \leftarrow i + 3$;
end
return (x);
```

Sample while loops

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 7$;
while ($i \leq n^2$) do
    $x \leftarrow x + i$;
    $i \leftarrow i + \lceil \sqrt{n} \rceil$;
end
return (x);
```

Sample while loops

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 1$;
while ($i \leq n$) do
    $x \leftarrow x + i$;
    $i \leftarrow 2 * i$ ;                    /* Note: Multiplication */
end
return (x);
```

Sample while loops

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 7$;
while ($i \leq n$) do
    $x \leftarrow x + i$;
    $i \leftarrow 2 * i$;
end
return (x);
```

Sample while loops

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 1$;
while $(i \leq n)$ do
    $x \leftarrow x + i$;
    $i \leftarrow 3 * i$;
end
return (x);
```

Sample loops

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    $j \leftarrow 1$;
    while $(j \leq n)$ do
        $x \leftarrow x + (i - j)$;
        $j \leftarrow 2 * j$;
    end
end
return (x);
```

Sample loops

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    $j \leftarrow 1$;
    while $(j \leq i)$ do
        $x \leftarrow x + (i - j)$;
        $j \leftarrow 2 * j$;
    end
end
return (x);
```

Sample loops (II)

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    $j \leftarrow 1$;
    while $(j \leq i)$ do
        $x \leftarrow x + (i - j)$;
        $j \leftarrow 2 * j$;
    end
end
return (x);
```

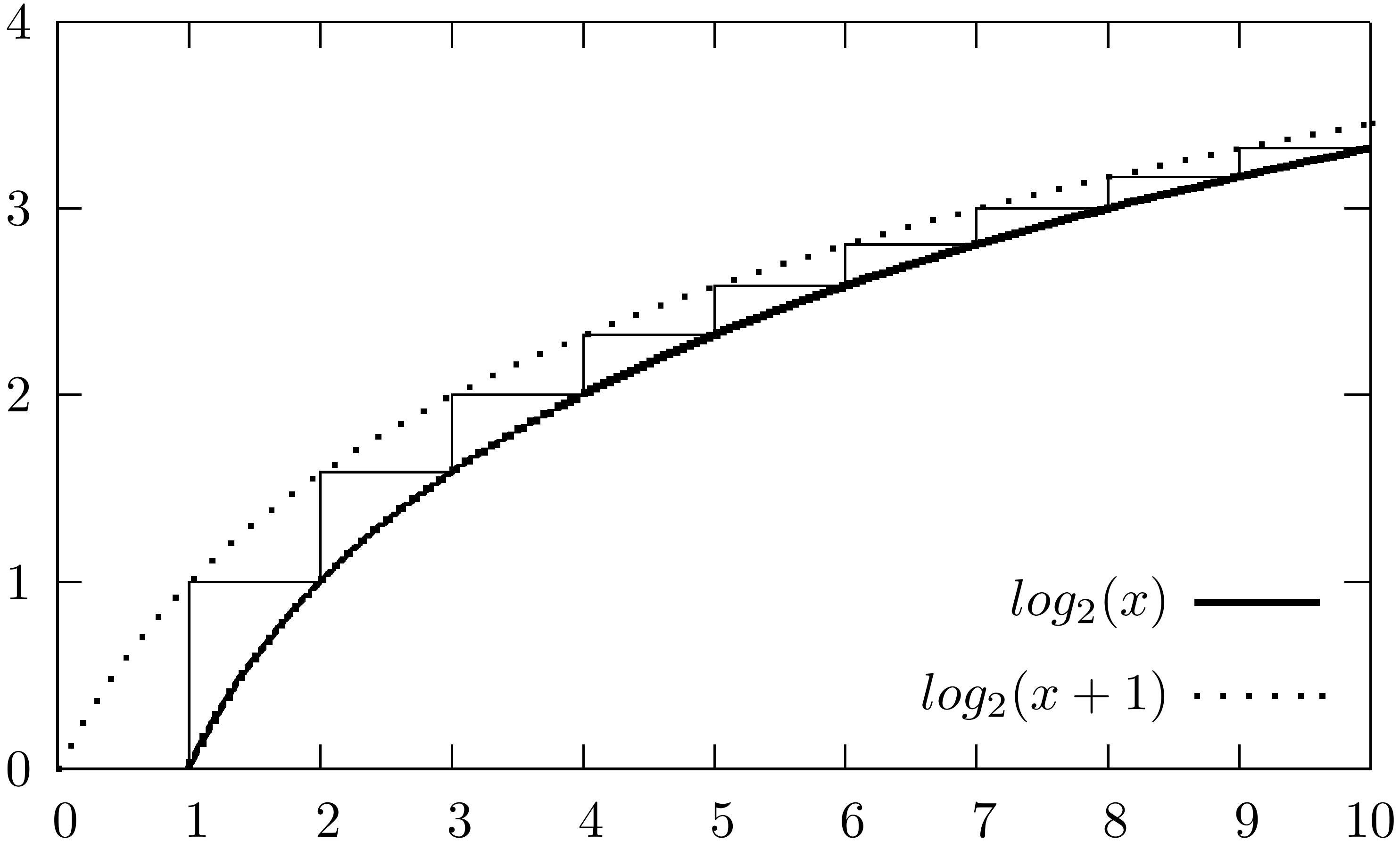
Evaluating $\sum f(i)$

To evaluate $\sum_{i=1}^{n} f(i)$:

- Compute upper and lower bounds on $\sum_{i=1}^{n} f(i)$.
- Use integrals (if possible):

$$\sum_{i=1}^{n} f(i) \approx \int_{1}^{n} f(x) dx.$$

Bounding a sum using integrals



Nested for loops

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $i$ do
        for $k \leftarrow j$ to $i$ do
            $x \leftarrow x + (k * i - j)$;
        end
    end
end
return (x);
```

Nested for loops (II)

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $i$ do
        for $k \leftarrow j$ to $i$ do
            $x \leftarrow x + (k * i - j)$;
        end
    end
end
return (x);
```

Nested loops

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 1$;
while ($i \leq n$) do
    for $j \leftarrow 1$ to $i$ do
        $x \leftarrow x + (i - j)$;
    end
    $i \leftarrow 2 * i$;
end
```

Nested loops (II)

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 1$;
while $(i \leq n)$ do
    for $j \leftarrow 1$ to $i$ do
        $x \leftarrow x + (i - j)$;
    end
    $i \leftarrow 2 * i$;
end
```

Nested loops

```
function func($n$)
    $x \leftarrow 0$;
    $i \leftarrow 1$;
    while $(i \leq n)$ do
        for $j \leftarrow 1$ to $i$ do
            $x \leftarrow x + (i - j)$;
        end
        $i \leftarrow 3 * i$;
    end
```

Nested loops

```
function func($n$)
$x \leftarrow 0$;
$i \leftarrow 1$;
while $(i \leq n)$ do
    for $j \leftarrow 1$ to $\lfloor n/i \rfloor$ do
        $x \leftarrow x + (i - j)$;
    end
    $i \leftarrow 2 * i$;
end
```

Geometric Series

$$1 + 1/2 + 1/2^2 + 1/2^3 + 1/2^4 + \ldots = 1/(1 - 1/2) = 2.$$

$$1 + 1/3 + 1/3^2 + 1/3^3 + 1/3^4 + \ldots = 1/(1 - 1/3) = 3/2.$$

$$1 + 2/3 + (2/3)^2 + (2/3)^3 + (2/3)^4 + \ldots = 1/(1 - 2/3) = 3.$$

For r where $0 \leq r < 1$,

$$1 + r + r^2 + r^3 + r^4 + \ldots = \frac{1}{1 - r}.$$

Sample for loops

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 2n$ to $(3n^2 + 5n)$ do
    for $j \leftarrow 1$ to $(i^3 + i^2)$ do
        $x \leftarrow x + (i - j)$;
    end
end
return (x);
```

Sample for loops (II)

```
function func($n$)
$x \leftarrow 0$;
for $i \leftarrow 2n$ to $(3n^2 + 5n)$ do
    for $j \leftarrow 1$ to $(i^3 + i^2)$ do
        $x \leftarrow x + (i - j)$;
    end
end
return (x);
```