BIG O NOTATION

Time-complexity

IMPORTANCE

precise vocabulary to talk about code performance

useful for discussing tradeoffs between different approaches

can identify which part is inefficient

more important than you think

EXAMPLE

Variation 1

```
function addUpTo(int n)
{
    int total = 0, count;
    for(count=1;count<=n;count++){
        total+=1;
    }
    return total;
}
```

Variation 2

```
function addUpTo(int n)
{
    return n*(n+1)/2;
}
```

BIG O NOTATION

- formal general estimate
- formally talks about run time of an algorithm’s growth as its input grows

$O(f(n))$

$f(n) = n$	$O(n)$ = **linear**
$f(n) = n^2$	$O(n^2)$ = **quadratic**
$f(n) = 1$	$O(1)$ = **constant**
$f(n) = \log n$	$O(\log n)$ = **more complex**

BIG O SHORTHANDS

1. Constants don’t matter

instead of	simply
O(2n)	O(n)
O(500)	O(1)
O(3n + 5)	O(n)

BIG O SHORTHANDS

2. Smaller terms don’t matter

instead of	simply
$O(2n + 5)$	$O(n)$
$O(n + 500)$	$O(n)$
$O(n^2 + 3n + 1)$	$O(n^2)$

BIG O SHORTHANDS

3. Arithmetic is constant time

```
function addUpTo(int n)
{
   return n*(n+1)/2;
}
```

BIG O SHORTHANDS

4. Variable assignment is constant

```
function returnZero(int n)
{
   int total = 0;
   char data = ‘C’;
   float deci = 1.234;

   return total;
}
```

BIG O SHORTHANDS

5. Array or Object element access is constant

```
function arrayAccess(int *array)
{
   int = catch;
   catch = array[3];
   return catch;
}
```

BIG O SHORTHANDS

6. **Length of loop is multiplied by complexity of whatever happens in the loop**

```
function addUpTo(int n)
{
    int total = 0, count;
    for(count=1;count<=n;count++){
        total+=1;
    }
    return total;
}
```

BIG O SHORTHANDS

6. **Length of loop is multiplied by complexity of whatever happens in the loop.**

```
/*code snippet */
   int total=0,count,flag;
   for(count=1;count<=5;count++){
      for(flag=1;flag<=3;flag++){
         total+=1;
      }
   }
```

BIG O SHORTHANDS

6. **Length of loop is multiplied by complexity of whatever happens in the loop.**

```
for(count=1;count<=5;count++){}          O(1)

for(count=1;count<=n;count++){}          O(n)
```

Thank You

For your attention