

Algorithm performance



Big-O: consecutive code



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)
by Christine Alvarado, Mia Minnes, and Leo Porter, 2015.

By the end of this video you will be able to...

- Combine the runtimes of smaller code snippets to analyze the performance of more complicated code

```
public static void reduce (int[] vals) {  
    int minIndex = 0;  
    for (int i=0; i < vals.length; i++) {  
        if (vals[i] < vals[minIndex] ) {  
            minIndex = i;  
        }  
    }  
    int minVal = vals[minIndex];  
    for (int i=0; i < vals.length; i++) {  
        vals[i] = vals[i] - minVal;  
    }  
}
```

```
public static void reduce (int[] vals) {  
    int minIndex = 0;  
    for (int i=0; i < vals.length; i++) {  
        if (vals[i] < vals[minIndex] ) {  
            minIndex = i;  
        }  
    }  
    int minVal = vals[minIndex];  
    for (int i=0; i < vals.length; i++) {  
        vals[i] = vals[i] - minVal;  
    }  
}
```

IVQ: sample run

```
public static void reduce (int[] vals) {  
    int minIndex =0;  
    for (int i=0; i < vals.length; i++) {  
        if (vals[i] < vals[minIndex] ) {  
            minIndex = i;  
        }  
    }  
    int minVal = vals[minIndex];  
    for (int i=0; i < vals.length; i++) {  
        vals[i] = vals[i] - minVal;  
    }  
}
```

```
public static void reduce (int[] vals) {
```

```
    int minIndex = 0; O(1)
```

```
    for (int i=0; i < vals.length; i++) {  
        if (vals[i] < vals[minIndex] ) {  
            minIndex = i;  
        }  
    }
```

```
    int minVal = vals[minIndex]; O(1)
```

```
    for (int i=0; i < vals.length; i++) {  
        vals[i] = vals[i] - minVal;  
    }
```

```
}
```

```
for (int i=0; i < vals.length; i++) {  
    if (vals[i] < vals[minIndex] ) {  
        minIndex = i;  
    }  
}
```

```
for (int i=0; i < vals.length; i++) {  
    if (vals[i] < vals[minIndex] ) {  
        minIndex = i;  
    }  
}
```



```
for (int i=0; i < vals.length; i++) {  
    if (vals[i] < vals[minIndex] ) {  
        minIndex = i;  
    }  
}
```

O(n)

```
public static void reduce (int[] vals) {
```

O(1)

```
    for (int i=0; i < vals.length; i++) {  
        if (vals[i] < vals[minIndex] ) {  
            minIndex = i;  
        }  
    }
```

O(n)

O(1)

```
    for (int i=0; i < vals.length; i++) {  
        vals[i] = vals[i] - minVal;  
    }
```

```
}
```

```
public static void reduce (int[] vals) {
```

$O(1)$

$O(n)$

$O(1)$

```
    for (int i=0; i < vals.length; i++) {  
        vals[i] = vals[i] - minVal;  
    }
```

```
}
```

```
for (int i=0; i < vals.length; i++) {  
    vals[i] = vals[i] - minVal;  
}
```

```
for (int i=0; i < vals.length; i++) {  
    vals[i] = vals[i] - minVal;  
}
```

```
for (int i=0; i < vals.length; i++) {  
    vals[i] = vals[i] - minVal;  
}
```

O(n)

```
public static void reduce (int[] vals) {
```

$O(1)$

$O(n)$

$O(1)$

```
    for (int i=0; i < vals.length; i++) {  
        vals[i] = vals[i] - minVal;
```

$O(n)$

```
    }
```

```
}
```

```
public static void reduce (int[] vals) {
```

$O(1)$

$O(n)$

$O(1)$

$O(n)$

```
}
```

Total: $O(n)$