# Assignment 5 - Run Times

Loic Konan

---

- Count = **1024 * 1024 = 1048576**
- Complexity = **O(n^2)**

```
int count = 0;
int n = 1024;
for(int i = 0; i < n; i++)
{
    for(int j = 0; j < n; j++)
    {
        count++;
    }
}
cout << count << endl;
```

---

- Count = **[1024 * (1024 - 1)] / 2 = 523776**
- Complexity = **O(n^2)** which was really **O[(n^2)/2]** but we just drop the constant(s).

```
int count = 0;
int n = 1024;
for(int i = 0; i < n; i++)
{
    for(int j = 0; j < i; j++)
    {
        count++;
    }
}
cout << count << endl;
```

---

- Count = **[1024^2 * (1024 - 1)] / 2 = 536346624**
- Complexity = **O(n^3)** which was really **O[(n^3)/2]** but we just drop the constant(s).

```
int count = 0;
int n = 1024;
for(int i = 0; i < n; i++)
{
    for(int j = 0; j < i; j++)
```

```
    {
        for(int k = 0; k < n; k++)
        {
            count++;
        }
    }
}
cout << count << endl;
```

---

- Count = **2 * 1024 = 2048**
- Complexity = **O(n)** which was really **O(2n)** but we just drop the constant(s).

```
int count = 0;
int n = 1024;
for(int i = 0; i < 2n; i++)
{
    count++;
}
cout << count << endl;
```

---

- Count = **2 * 1024 * 1024 = 2097152**
- Complexity = **O(n^2)** which was really **O[2(n^2)]** but we just drop the constant(s).

```
int count = 0;
int n = 1024;
for(int i = 0; i < n; i++)
{
    for(int j = 0; j < 2*n; j++)
    {
        count++;
    }
}
cout << count << endl;
```

---

- Count = **1024 + [1024 * (1024 - 1)] / 2 = 524800**
- Complexity = **O(n^2)** which was really **O[n + (n^2 / 2)]** but we just drop the constant(s).

```
int count = 0;
int n = 1024;
for(int i = 0; i < n; i++)
```

```
{
    count++;
}
for(int j = 0; j < n; j++)
{
    for(int k = 0; k < j; k++)
    {
        count++;
    }
}
cout << count << endl;
```

---

- Count = **1024 * [(log 1024) + 1] = 11264**
- Complexity = **O(n log n)** which was really **O[n * ((log n) + 1) ]** but we just drop the constant(s).

```
int count = 0;
int n = 1024;
int i = n;
while(i > 0)
{
    for(int j = 0; j < n; j++)
    {
        count++;
    }
    i /= 2;
    cout << i << endl;
}
cout << count << endl;
```

---

- Comparisons = **log 1024 = 10**
- Complexity = **O(log n)**

```
bool found = 0;
int n = 1024;
int i = n;
// assume loaded with random numbers
// and in ascending order.
int *A = new int[n];

// Whats most number of comparisons ?
// Whats the complexity ?
found = BinarySearch (A, n);
```

---

- Count = **0**
- Complexity = **O(log n)** which was really **O[(log n) + 2]** but we just drop the constant(s).

```
int count = 0;
int n = 1024;
int i = n;
while(i > 0)
{
    cout << i << endl;
    i /= 2;
}
cout<<count<<endl;
```