

Big-O-Notation Analysis

1. ignore constants
2. certain terms dominate others

name: Kip Goney

Part A:

```
void f1(int n)
```

```
{
```

```
  int i=2;
```

```
  while(i < n){
```

```
    // O(1)
```

```
    i = i * i;  → exponential growth
```

```
  }
```

```
}
```

$n = i_0 \cdot i_1 \cdot \dots \cdot i_x$ where x is the # of loops
 $i_0 = 2$, so, ... $\log_2(n) = x$
 $\log(\log(n)) = x$ iterations through the

$O(\text{runtime}) = \Theta(\log(\log(n)))$
 $\Omega(\text{runtime}) = \Omega(\log(\log(n)))$

thus $\log(\log(n))$

$\sum_{x=0} \Theta(1) = \Theta(1 \cdot \log(\log(n)))$

final answer

$= \Theta(\log(\log(n)))$
for the while loop

Part B:

```
void f2(int n){
```

```
  for(int i=1; i <= n; i++){
```

```
    if((i % (int)sqrt(n)) == 0){
```

```
      for(int k=0; k < pow(i,3); k++){
```

```
        // O(1) run-time
```

```
      }
```

```
    }
```

```
}
```

outer loop runtime: $i < n$;

if statement run-time: only runs if i is divisible by \sqrt{n} . so combined

inner loop: $k < i^3$; $\Theta(n^3)$

Final Answer

$$\Theta(\sqrt{n} \cdot n^3) = \Theta(n^{\frac{1}{2}} \cdot n^{\frac{6}{2}}) = \Theta(n^{\frac{7}{2}})$$

} the run time for the outer loop if statement will be $\Theta(\frac{n}{\sqrt{n}})$ which is really just $\Theta(\sqrt{n})$

Part C:

```

for (int i = 1; i <= n; i++) {
    for (int k = 1; k <= n; k++) {
        if (A[k] == i) {
            for (int m = 1; m <= n; m = m + m) {
                // do something for O(1) runtime
            }
        }
    }
}

```

Annotations:

- $\rightarrow O(n)$ for the first loop.
- $\rightarrow O(n)$ for the second loop.
- $\rightarrow O(1)$ for the if statement (because it's on an if statement).
- $\rightarrow O(\log(n))$ for the third loop (essentially doubling n every time).
- $\rightarrow O(1)$ for the inner loop (do something for $O(1)$ runtime).

Solution: $\sum \theta_{\text{runtime}} = \theta(n \cdot n \cdot 1 \cdot \log(n) \cdot 1) = \theta(n^2 \cdot \log(n))$

Outer Loop: $i \leq n, i++$. So $\theta(n)$
 2nd outmost loop: $k \leq n, k++$. So $\theta(n)$
 if-statement: Only runs if $A[k] = i$
 inner-loop: $m \leq n; m = m + m$; so $\theta(\log(n))$

$O = O(n^2 \log(n))$
 $\Omega = \Omega(n^2)$



Thus,

Final Answer:
 $\theta_{\text{run-time}} \Rightarrow \theta(n^2)$

Part D:

```

int f (int n)
{
    int *a = new int [10];
    int size = 10;
    for (int i = 0; i < n; i++)
    {
        if (i == size)
        {
            int newSize = 3 * size / 2;
            int *b = new int [newSize];
            for (int j = 0; j < size; j++) b[j] = a[j];
            delete [] a;
            a = b;
            size = newSize;
        }
        a[i] = i + 1;
    }
}

```

outer loop: $\sum_{i=0}^n \theta(1) = \theta(n)$

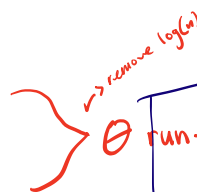
if statement: runs if $i = \text{size}$ $(\frac{3}{2})^m$, so $\theta(\log(n))$ for $i = \log(n)$ (takes precedence)
 inner-loop: occurs for $\theta(n (\frac{3}{2})^{\log(n)})$

run-time

$\Omega(n)$ \leftarrow run-time for $n < 10$

$O(n \cdot \log(n)) = (n \log(n))$

\hookrightarrow might be $O(n^2 \log(n))$



Final Answer

$\theta_{\text{run-time}} = \theta(n)$

?? $\theta(\log(n))$??