

Part (A)

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

```
{
```

```
    int i=2;  $\Theta(1)$ 
```

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

```
        /* do something that takes  $O(1)$  time */
```

```
        i = i*i;
```

```
    }
```

```
}
```

Part 1
Part 2

Part 1: $\Theta(1)$

Part 2: $\Theta(\log n)$, as i doubles in size each loop, so it will take $\log n$ loops for i to be $\geq n$

$\Theta(\log n)$

Part (B)

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

```
{
```

```
    for(int i=1; i <= n; i++) { Part 3
```

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

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

```
                /* do something that takes O(1)
```

```
            time */
```

```
        }
```

```
    }
```

```
}
```

```
}
```

Part 1: $\Theta(i^3)$, as k must be $\geq \text{pow}(i, 3)$ before the loop ceases

Part 2: $\Theta(\sqrt{n})$, as the if condition will be true only \sqrt{n} times,
as i must be a multiple of \sqrt{n} , and $\sqrt{n} \cdot \sqrt{n} = n$,
so the final loop $i = n$.

Part 3: $\Theta(n)$, as i must be $\geq n$ to stop the for loop

$$\text{Part 1} \rightarrow \text{Part 2: } n\sqrt{n} \sum_{i=1}^{\sqrt{n}} i^3 \approx n\sqrt{n} \left((\sqrt{n})^3 + (\sqrt{n}-1)^3 + \dots + 1 \right)$$

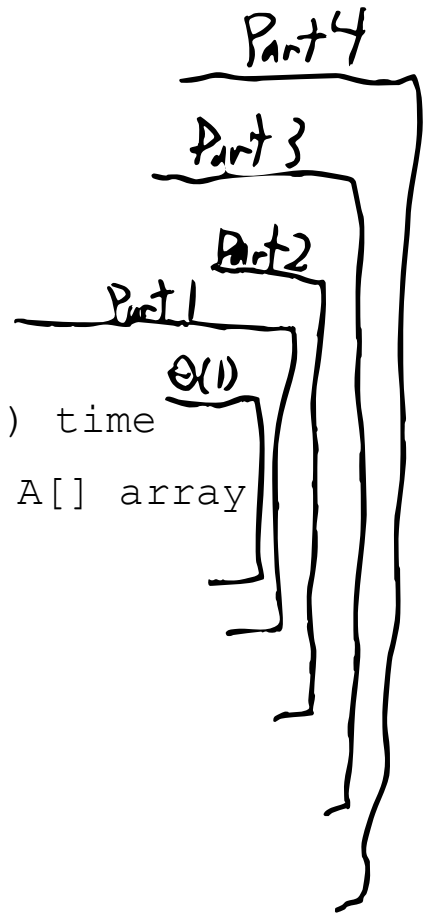
Part 2 \rightarrow Part:

$$\approx \frac{3}{2} n^{\frac{3}{2}} = \frac{3}{2} n\sqrt{n} \approx \Theta(n^{\frac{3}{2}})$$

$$\Theta(n + n^{\frac{3}{2}}) = \boxed{\Theta(n^{\frac{3}{2}})}$$

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 that takes O(1) time  
                // Assume the contents of the A[] array  
                are not changed  
            }  
        }  
    }  
}
```



Part 1: $\Theta(\log n)$, as m doubles in size each loop

Part 2: $\Theta(n)$, as worst case scenario every element in A is equal to i ,

Part 3: $\Theta(n)$, as k must reach n

Part 4: $\Theta(n)$, as i must reach n

which means
the if block
is called n
times

$$\Theta(n \cdot n + n \log n) = \boxed{\Theta(n^2)}$$

$$\text{Runtime} = \frac{\# \text{ of actions}}{\# \text{ of step}}$$

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*i;
```

```
    }
```

```
}
```

Step

Actions

1-9

9

10

10

11-14

4

15

15

16-21

7

...

...

The # of steps is increases alongside the # of actions, therefore the runtime inside the for loop is essentially $\Theta(1)$ for large n

5 7 11 17 ...
10 15 22 33 44 73 109 163 244

$$\Theta(n+1) = \boxed{\Theta(n)}$$