

Aaron Shields

a) void f1(int n)

int i = 2

while (i < n) {

i = i \* 1;

}

}

$$i \cdot 2 = O(\log_2 n)$$

$$i \cdot 3 = O(\log_3 n)$$

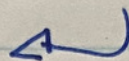
$$k \cdot k = k^2$$

$$k^2 \cdot k^2 = k^4$$

$$k^{\sqrt{n}} = n$$

$$\sqrt{n}$$

$$k^2 = O(\sqrt{n})$$



b) void f2(int n)

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

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

for (int k = 0; k <= pow(1, 3); k++) {

}

$$1^3 = 1$$

}

}

}

$O(n)$  - Outer loop

$O(\sqrt{n})$  - inner

$O(\sqrt{n}^3)$  -

$$O(n) + O(\sqrt{n}) + O(\sqrt{n}^3)$$

$$\text{time complexity} = O(n\sqrt{n})$$

c) for (int i = 1; i <= n; i++)

$$O(n) \quad n + n + n + n$$

for (int k = 1; k <= n; k++)

$$O(n) \quad k + k + k + \dots$$

if (A[k] == i)

for (int m = 1; m <= n; m = m \* m) {

$$O(\log n)$$

// O(1)

}

$$n^2 \log n = \text{time complexity}$$

$$n^2 \log(n)$$

d) int \* a = new int[10];

int size = 10;

for (int i = 0; i < size; i++) {

if (i == size)

int newSize = 3 \* size / 2

int \* b = new int[newSize];

for (int j = 0; j < size; j++) {

a[j] = b[j];

a = b

a[j] = 1 + 1;

$$O(\text{size})$$

$$O(n-1) + O(n)$$

$$O(n-1)$$

$$O(n)$$

$$\text{time complexity} = O(n)$$