Algorithmic Analysis I: Group Exercises

CSCI 246

March 13, 2026

Problem 1. Consider the following code block:

```
def f(n):
  x = 5, y = 8;
  z = x + y;
  return z + n
```

Elemenatry operations include assignments, arithmetic operations, comparisons, and function returns.

A. What is the number of elemenatry operations of f (denoted as $T(n)$) as a function of the input n?

$$T(n) = 6$$

B. What is the asymptotic behavior of f (i.e., use big-O, big-Omega, and/or big-Theta notation)?

The function f runs in constant time $\Theta(1)$.

Problem 2. Consider the following code block:

```
def f(n):
  s = 0;                  # 1 operation
  for i = 0 to n - 1:     # 1 assignment/increment & 1 comparison for each iteration
    s = s + 1             # 2 operations per iteration
  return s                # 1 operation
```

Elemenatry operations include assignments, arithmetic operations, comparisons, and function returns.

A. What is the number of elemenatry operations of f (denoted as $T(n)$) as a function of the input n?

$T(n) = 4n + 2$ $\quad$ $2n/3n/4n$ would all be acceptable.

B. What is the asymptotic behavior of f (i.e., use big-O, big-Omega, and/or big-Theta notation)?

The function f runs in linear time $\Theta(n)$.

Problem 3. Consider the following code block:

```
def f(n):
  a = new array[n];     # n time to allocate a new array

  for i = 0 to n - 1:   # 2 operations per iteration; n iterations
    a[i] = i + 1        # 2 operatrions per iteration

  s = 0;                # 1 operation
  for i = 0 to n - 1:   # 2 operations per iteration; n iterations
    s = s + a[i]        # 3 operations per iteartion
  return s              # 1 operation
```

Elemenatry operations include assignments, arithmetic operations, comparisons, and function returns.

A. What is the number of elemenatry operations of f (denoted as $T(n)$) as a function of the input n?

$$T(n) = 10n + 2$$

B. What is the asymptotic behavior of f (i.e., use big-O, big-Omega, and/or big-Theta notation)?

The function f takes linear time to run ($\Theta(n)$).

Problem 4. Consider the following code block:

```
def insert(a, i):
  e = a[i];
  while 0 < i:
    if a[i - 1] < e:
      a[i] = a[i - 1];
      i = i - 1
    else:
      break
  a[i] = e
```

```
def sort(a):
  n = length a;
  for i = 1 to n - 1:
    insert(a, i)
```

Elemenatry operations include assignments, arithmetic operations, comparisons, and function returns.

A. What is the number of elemenatry operations performed by insert (denoted as $T(n, i)$) as a function of n (the size of a) and i?

$$T(n, i) = 3 \text{ (Best Case)} \qquad T(n, i) = 7i + 3 \text{ (Worst Case)}$$

B. What is the number of elemenatry operations performed by sort (denoted as $T(n)$) as a function of of n (the size of a)?

$$T(n) = 2 + \sum_{i=1}^{n-1} 2 + 3 = 5(n-1) + 2 = 5n - 3 \text{ (Best Case)}$$

$$T(n) = 2 + \sum_{i=1}^{n-1} 2 + 7i + 3 = \frac{7n^2 + 3n}{2} - 3 \text{ (Worst Case)}$$

C. What is the asymptotic behavior of sort (i.e., use big-O, big-Omega, and/or big-Theta notation)?

In the best case (when a is reverse sorted) sort runs in linear time ($\Theta(n)$). In the worst case (a is sorted) and in the average case, sort runs in quadratic time ($\Theta(n^2)$).