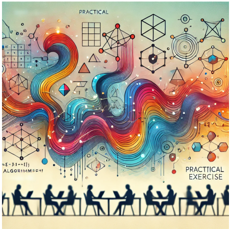

THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

# Algorithms and Data Structures

## Algorithm Analysis: Group Discussion Questions

Jiamou Liu
The University of Auckland




THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

**Question 1.** Consider the following code snippet and analyze its running time. Come up with a function $f(n)$ such that the algorithm runs in $\Theta(f(n))$.

```
for i = 1 to n do
    for j = i to n do
        for k = i to j do
            constant_operation()
```

**Question 2.** Consider the following code snippet and analyze its running time. Come up with a function $f(n)$ such that the algorithm runs in $\Theta(f(n))$.

```
for (i = 1; i ≤ n; i = i + 1) do
    for (j = 1; j ≤ i; j = j × 2) do
        for (k = i; k ≤ n; k = k + j) do
            constant_operation()
```

**Question 3.** The Euclidean algorithm computes the greatest common divisor (GCD) of two positive integers $a$ and $b$:

1. Given two integers $a$ and $b$ (where $a \geq b$), compute $r = a\%b$.
2. Replace $a$ with $b$ and $b$ with $r$.
3. Repeat steps 1 and 2 until $b = 0$. When this happens, $a$ is the GCD of the original inputs.

The pseudocode for this algorithm is as follows:

```
function EUCLIDEAN_GCD(a, b)
    while b ≠ 0 do
        r ← a%b          (comment: taking the remainder of a ÷ b)
        a ← b
        b ← r
    return a
```

- What is the worst-case input pair (or sequence) that maximizes the number of steps?
- Derive the worst-case running time of this algorithm in terms of the number of divisions required, expressed using the *value of the inputs* $n = \max\{a, b\}$.

**Question 4.** Consider a hash table $T$ of size $m$ where we use *linear probing* to resolve collisions. Each key $k$ is hashed to an index $h(k)$ in the range $[0, m-1]$. If a collision occurs (i.e., the slot $h(k)$ is already occupied), the algorithm checks the next slot $(h(k)+1)\%m$, then $(h(k)+2)\%m$, and so on, until an empty slot is found.

Assume that:

- The hash table has a load factor $\alpha = \frac{n}{m}$, where $n$ is the number of keys inserted and $m$ is the total number of slots in the table.
- The hash function distributes keys uniformly, so each position in the hash table is equally likely to be chosen for a new key.

The pseudocode for inserting a key $k$ with linear probing is as follows:

```
function HASH_INSERT(T, k)
    i ← 0
    repeat
        j ← (h(k) + i)%m
        if T[j] = NIL then
            T[j] ← k
            return j
        else
            i ← i + 1
```

1. Describe the worst-case scenario for insertion with linear probing. You may express the running time as a function of $m$. Explain your answer.
2. Assume $\alpha < 1$ (i.e., the hash table is not completely full). Define the average case as the expected number of probes required to insert a new key into the hash table with $\alpha$ as the load factor. Derive an expression for the expected number of probes required to find an empty slot in terms of $\alpha$.