INT 102 Algorithmic Foundations

Problem Session 1

1. Give the trace table and the output of the following algorithm for m = 16.

```
input m
count = 0
x = 1
while x < m do
    begin
        x = x ∗ 2
        count = count + 1
    end
output count
```

What is the output for general m that is a positive power of 2 (i.e., m = 2, 4, 8, 16, 32, . . .)?

Solution:

The trace table for m = 16.

	x	count
Before while loop	1	0
1st iteration	2	1
2nd iteration	4	2
3rd iteration	8	3
4th iteration	16	4

The output of the algorithm for m = 16 is 4.

In general, the algorithm computes the value $\log_2 m$.

2. Rewrite the following for-loop into a while loop.

```
input m
x = 0
for i = 1 to m do
  begin
     x = x + i
  end
output x
```

Solution:

```
input m
x = 0, i = 1
while i <= m do
   begin
      x = x + i
      i = i + 1
   end
output x
```

3. Rewrite the above for-loop into a repeat loop.

   Solution:

```
input m
x = 0, i = 1
if m >= 1 then
      begin
        repeat
           x = x + i i = i + 1
        until i > m
      end
output x
```

4. List the following functions from lowest order to highest order of magnitude:

   $O(1)$, $O(n^2)$, $O(n^3)$, $O(n)$, $O(\log n)$, $O(\log^2 n)$, $O(n \log n)$, $O(\sqrt{n})$, $O(n^2 \log^4 n)$, $O(n^{3/2})$, $O(n^{10/3})$, $O(n^4)$.

   Solution:

   The order is:
   $O(1)$, $O(\log n)$, $O(\log^2 n)$, $O(\sqrt{n})$, $O(n)$, $O(n \log n)$, $O(n^{3/2})$, $O(n^2)$, $O(n^2 \log^4 n)$, $O(n^3)$, $O(n^{10/3})$, $O(n^4)$.

   NOTE: It is NOT correct to say that $1 < \log n < \log^2 n < \cdots$.
   It is only correct to say that 1 is of lower order than log n, log n is of lower order than log2 n, etc., because we focus on large n when we talk about order.

5. Prove that the function $f(n) = 3n^2 \log n + 2n^3 + 3n^2 + 4n$ is $O(n^3)$.
(That is, show that there exists constants c and n_0 such that $f(n) \leqslant cn^3$ for all $n > n_0$.)

Solution:
To prove that $3n^2$ log n + 2n3 + 3n2 + 4n is O(n3), we show that there exists a constant c and n0 such that for any integer n ⩾ n0,
3n2 log n+2n3 log2 n+3n2+4n ⩽ c n3.

- 3n2 log n ⩽ 3n3 ∀n ⩾ 1,
- 2n3 ⩽ 2n3 ∀n,
- 3n2 ⩽ 3n3 ∀n ⩾ 1, and
- 4n ⩽ 4n3 ∀n ⩾ 1.

As a result, 3n2 log n + 2n3 log2 n + 3n2 + 4n ⩽ 12n3 ∀n ⩾ 1.
Since 12 is a constant, the function 3n2 log n + 2n3 log2 n + 3n2 + 4n is O(n3).
N.B.: Should give reasons for the inequalities as shown above.
Alternatively, we can prove in the following way:

- 3n2 log n ⩽ n3 ∀n ⩾ 16,
- 2n3 ⩽ 2n3 ∀n,
- 3n2 ⩽ n3 ∀n ⩾ 3, and
- 4n ⩽ n3 ∀n ⩾ 2.

As a result, 3n2 log n + 2n3 log2 n + 3n2 + 4n ⩽ 5n3 ∀n ⩾ 16.

6. Consider a string T of n characters T[0..(n-1)]. Design and write a pseudo code of an algorithm to determine if a given character, denoted by C, appears uniquely in T [0..(n-1)], i.e., whether the character C appears exactly once in T.

For example, if T [ ] is “Hello, how are you?" and C is `H', then the algorithm should report “Yes, the character appears uniquely.". However, if C is `e', then the algorithm should report “No, the character does not appear uniquely.

Suggested Solution:

```
Algorithm:    CC1(c, T[0..n-1])
// determine if a given character, denoted by C, appears uniquely in T [0..(n-1)],
//Input:    an array of n numbers T[0, n-1]
//Output: “Yes” if c appears uniquely in T [0..(n-1)], otherwise “No”
      counter = 0
      for i = 0 to n-1
         if c=T[i] then
            counter = counter + 1
      if counter = 1 then
            return “Yes”
      else
            return “No”
```

or

```
Algorithm:    CC2(c, T[0..n-1])
// determine if a given character, denoted by C, appears uniquely in T [0..(n-1)],
//Input:    an array of n numbers T[0, n-1]
//Output: “Yes” if c appears uniquely in T [0..(n-1)], otherwise “No”
      counter = 0
      while counter < 2    do
          if c=T[i] then
             counter = counter + 1
      if counter = 1 then
             return “Yes”
      else
             return “No”
```

Problems for Fun (Optional):

- [Puzzle] A farmer is standing on the left side of the river and with him are a wolf, a goat and a box of cabbages. In the river there is a small boat. The farmer wants to cross the river with all the three items that are with him. There are no bridges and in the boat there is only room for the farmer and one item. If he leaves the goat with the cabbages alone on one side of the river the goat will eat the cabbages. If he leaves the wolf and the goat on one side the wolf will eat the goat. How can the farmer cross the river with all three items, without one eating the other?

  Suggested Solution:

  The farmer can cross the river as follows. Notations: F-farmer, W-wolf, G-goat, C-cabbages. Remind that W&G and G&C cannot be left alone without F.

	left bank	boat	direction	right bank
initial	W,G,C	—	—	nothing
1st ride	W,C	F+G	→	nothing
2nd ride	W,C	F	←	G
3rd ride	C	F+W	→	G
4th ride	C	F+G	←	W
5th ride	G	F+C	→	W
6th ride	G	F	←	W,C
7th ride	nothing	F+G	→	W,C
final	nothing	—	—	W,G,C

- [Puzzle] Four people are going to cross a bridge from the same side. It is night and they have only one flashlight. At each time there can be at most two people crossing the bridge. Any party that crosses the bridge, either one or two people, must have a flashlight to walk with them. The flashlight must be walked back and forth and cannot be thrown, for example. Each person walks at a different speed: person A takes 1 minute to cross the bridge, person B 2 minutes, person C 5 minutes and person D 10 minutes. A pair must walk together at the rate of the slower person's pace. For example, if person A and person D walk together, it takes 10 minutes to get to the other side of the bridge.

  How can they arrive the other side of the bridge in the shortest time? How many minutes does this take?

Suggested Solution:

A, B, C, and D take 1, 2, 5, and 10 minutes respectively.

	left side	walk	direction	right side	time taken
initial	A, B, C, D	—	—	nobody	
1st trip	C, D	A + B	→	nobody	2 mins
2nd trip	C, D	A	←	B	1 min
3rd trip	A	C + D	→	B	10 mins
4th trip	A	B	←	C, D	2 mins
5th trip	nobody	A + B	→	C, D	2 mins
final	nobody	—	—	A, B, C, D	total 17 mins

N.B.: 2nd trip can be B, and then 4th trip becomes A, total time is still 17 mins.

- [Puzzle] Twelve eggs look identical except one is of different weight (can be heavier or lighter). How can you weigh only four times on a balance scale to find out which one is different and whether it is heavier and lighter?

  Suggested Solution:

  Divide the eggs into four groups.

  - Weigh any of the two groups. If their weights are different, then the special egg is in these two groups; otherwise, the special egg is in the remaining two groups. In both cases, we know that two groups are normal and one of the remaining two groups contain the special egg.
  - Weigh a normal group with one of the potential special groups. If their weights are the same, we identify the special group; otherwise, the last group is the special group. Suppose the special group contains the eggs X, Y , and Z.

  - Pick two eggs X and Y from the special group and weigh against two normal eggs.
    a) If the weights are different, then either X or Y is the special egg and we also know by now whether the special egg is heavier or lighter; then weighing X against Y will tell us which is the special egg.
    b) Otherwise, Z is the special one and weighing Z with any other egg will tell us whether it is heavier or lighter.