# The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 4 MODULE, AUTUMN/SPRING SEMESTER 2021-2022

**ADVANCED ALGORITHMS AND DATA STRUCTURES**

Time allowed: TWO HOURS

---

*This is a take-home and open-book exam with answers to be submitted in Moodle no later than the DATE/TIME indicated in the Moodle dropbox. The Academic Integrity Statement and general Instructions and Advice are also available in Moodle.*

***Answer ALL FOUR QUESTIONS***

Submit your answers in PDF files, with each page in the correct orientation, to the dropbox in the module's Moodle page. Please produce one separate PDF file for each of the questions that you answer. This means that you will submit maximum 4 PDF files.

You are recommended to write/draw your answers on paper and then scan them to a PDF file. You may also type/draw your answers into electronic form directly and generate a PDF file, but do not lose time in typesetting or making pretty graphics.

Your solutions should include complete explanations and should be based on the material covered in the module. Make sure your PDF files are easily readable and does not require magnification. Text/drawing which is not in focus or is not legible for any other reason will be ignored.

Use the following naming convention for your PDF files: `StudentID_COMP4019_QX`, where `X` is replaced by the question number (1,2,3,4). Include your student ID number at the top of the first page in each PDF file. Please do not include your name.

Staff are not permitted to answer assessment or teaching queries during the period in which your examination is live. If you spot what you think may be an error on the exam paper, note this in your submission but answer the question as written.

You must produce the answers by yourself only. You must adhere to the University's Policy on Academic Integrity and Misconduct (`https://www.nottingham.ac.uk/studyingeffectively/studying/integrity/index.aspx`). You are also not allowed to share this exam paper with anyone else or post it anywhere online.

# QUESTION 1 – Complexity Classes, Recurrence Relations, Amortized Analysis [25 marks]

## 1.1 Big-O Formal Proof [7 marks]

Prove that $n^2 \in O(2^n)$ using the formal definition of $O$.

## 1.2 Mysterious Algorithm [10 marks]

Consider the following pseudocode of a recursive algorithm:

**function** mystery($f, l, r$)
  **if** $l = r$ **then**
    **return** $l$
  $l3 \leftarrow (2l + r)/3$
  $r3 \leftarrow (2r + l)/3$
  **if** $f(l3) < f(r3)$ **then**
    **return** mystery($f, l3, r$)
  **else**
    **return** mystery($f, l, r3$)

Assume that the call to $f(x)$ is done in $\Theta(1)$. Write mystery's recurrence relation and analyse its worst-case time complexity using the master theorem.

## 1.3 Amortized Complexity [8 marks]

An ArrayList is a data structure that implements the functionality of a List using an array as its underlying storage. An ArrayList $L$ has $n$ elements, a capacity $c$, and an array $array$ of size $L.c$ containing its elements. The pseudocode of the insert operation is given below. Assume that the function array($k$) allocates and returns an empty array of size $k$ in $\Theta(1)$.

**function** insert($L, x$)
  **if** $L.n = L.c$ **then**
    $L.c \leftarrow 2 \times L.c$
    $a \leftarrow$ array$(L.c)$
    **for** $i \in \{0, \ldots, L.n - 1\}$ **do**
      $a[i] \leftarrow L.array[i]$
    **end for**
    $L.array \leftarrow a$
  $L.array[L.n] \leftarrow x$
  $L.n \leftarrow L.n + 1$

What is insert's worst-case cost? What is its amortized cost? Justify your answers.

# QUESTION 2 – Dynamic Programming [35 marks]

Three goblin warchiefs, Wig, Brök, and Güg, wish to assault an enemy fortress by catapulting goblins over the their walls. The further the trebuchet is from its target, the better (to avoid retaliation and sorties from the defenders); however, goblins are fragile beings, and can only be in fighting shape up to a maximum distance $d^*$. The trebuchet itself can throw most goblins up to a distance of $D$. You can assume the following:

- All distances are measured in integer increments;
- A goblin thrown a distance $d \leq d^*$ will always be in fighting shape;
- A goblin thrown a distance $d > d^*$ will never be in fighting shape;
- It is possible that $d^* = 0$ or $d^* > D$;

Help the warchiefs determine the value of $d^*$ (or the fact that $d^* > D$) in as few throws as possible, while avoiding losing too many "volunteers"...

1. Wig does not want to waste perfectly good goblins unnecessarily. He only has $G = 1$ goblins to determine $d^*$. What is the only strategy that is guaranteed to find $d^*$ (or to determine that $d^* > D$), and what is its cost? **[3 marks]**
2. For Brök, goblins are a means to an end. He makes $G \geq D$ available to determine $d^*$ in as few throws as possible. What is the strategy that minimizes the number of throws, and what is its worst-case cost (justify)? **[4 marks]**
3. Finally, Güg wants to know the full picture before deciding of a strategy.
   (a) Let $t(g, k)$ be the worst case for the minimum number of throws, given $1 \leq g \leq G$ remaining goblins and $0 \leq k \leq D$ consecutive distances remaining to try. What are the base cases of $t(1, k)$ and $t(g, 0)$? **[2 marks]**
   (b) Write a recursive equation for $t(g, k)$. Justify. Hint: there are $k$ possible distances to try, and for each, there are two possible cases, depending on the result of the throw. **[12 marks]**
   (c) Describe why dynamic programming helps here. **[4 marks]**
   (d) Using your equation, calculate[1] $t(G, D)$ with $G = 2$ and $D = 7$. You may find it useful to copy and fill the following table. Start with filling in the base cases. **[5 marks]**

| $g$ \ $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | |
| 2 | | | | | | | | ? |

   (e) What is the complexity of computing $t(G, D)$ using your dynamic programming approach? **[5 marks]**

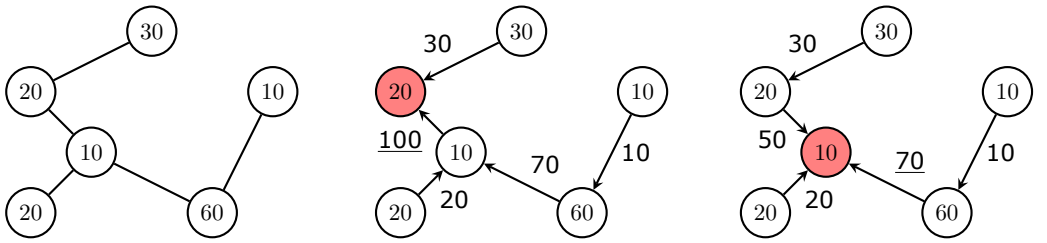
[1] Implementing $t$ is not necessary here; manually computing such a small instance should take no more than a minute or two.

# QUESTION 3 – Graphs [30 marks]

You are hired to help a town planner organize an event, while minimizing traffic in Nottingham. In this context, we model the city and its roads as an undirected graph $G = (V, E)$, where $V$ is a set of nodes (intersections or points of interest within the city) and $E$ is a set of edges (roads connecting the nodes). The graph is a tree, meaning there exists a single path between each pair of vertices. Let $n = |V|$. Since $G$ is a tree, we have[2] $|E| = n - 1$.

The event is to happen at a given node, that is to be determined. Each node $v \in V$ has a population number $p(v) \geq 0$, representing the number of people living in the catchment area of that node, and that will commute to and from the event. After the event, all participants will travel back to their original node. This will cause congestion on each road $e \in E$ that is equal to the total number of event-goers that travel along that road. The goal is to determine the ideal location of the event $v^* \in V$ that would minimize the maximal congestion.

Figure 1 below gives an example of $G$ and $p$, and shows the resulting congestion from choosing two different nodes as $v^*$.




(a) Graph $G$ with $p(v)$ shown inside nodes.

(b) Choosing the red node as $v^*$. Resulting maximum congestion is 100.

(c) Choosing the red node as $v^*$. Resulting maximum congestion is 70 (minimum for this particular $G$ and $p$).

Figure 1: Minimizing congestion.

1. Write a formal statement of the problem. **[5 marks]**
2. Does it make more sense to represent $G$ with an adjacency list or an adjacency matrix? Explain. **[3 marks]**
3. Describe a $\Theta(n^2)$ algorithm to find $v^*$. Justify its complexity. **[10 marks]**
4. Describe a $\Theta(n)$ algorithm to find $v^*$. Justify its complexity. **[12 marks]**

[2] Indeed, one more edge would introduce a cycle, and one less would disconnect the graph.

## QUESTION 4 – Miscellaneous [10 marks]

1. A typical AES (a symmetric-key encryption scheme) key is 128 to 256 bits. A typical RSA (a public-key encryption scheme) is 2048 to 8192 bits. Explain, in your own words, why this difference exists (from the point of view of algorithms!). **[5 marks]**

2. Explain why, among all the possible paths from the root of a red-black tree to any of its leaves, the longest path is at most twice a long as the shortest path. **[5 marks]**