ALGORITHMS & DATA STRUCTURES 2
Mid-Term Coursework
Where's Goldie?

1. INTRODUCTION

Where's Goldie? is a search game. But, instead of finding a person in a page filled with hundreds of drawings of people doing amusing things, you must a find a number (Goldie) in a **square matrix** of integers made of N rows and N columns. The range of the integers stored in the matrix goes from 0 to 500.

No number is repeated in the matrix. The number representing Goldie is entered as an input parameter (G) to the function FIND_GOLDIE(M, N, G), where M is the matrix, N the number of rows (or columns) and G the number to search for.

We have received 3 algorithmic proposals to do this. They are as follows.

Proposal 1: You must visit every element in the square matrix. If the element you are visiting is equal to G, then you have found Goldie! In that case, you return the position of Goldie as a pair (row, column). If, after visiting every element of the matrix, you did not find Goldie, you return the pair (-1,-1).

Proposal 2: Before visiting the elements of the square matrix, you process its content by sorting the elements of each row from smallest to largest. Figure 2 shows an example of the original matrix (left) and how its content is rearranged after sorting every row (right).

$$\begin{array}{c|ccccc|} & [0] & [1] & [2] & [3] & [4] \\ \hline [0] & 3 & 23 & 14 & 6 & 18 \\ [1] & 97 & 32 & 12 & 75 & 49 \\ [2] & 25 & 11 & 48 & 4 & 10 \\ [3] & 5 & 98 & 7 & 21 & 31 \\ [4] & 1 & 46 & 34 & 9 & 17 \\ \hline \end{array} \Rightarrow \begin{array}{c|ccccc|} & [0] & [1] & [2] & [3] & [4] \\ \hline [0] & 3 & 6 & 14 & 18 & 23 \\ [1] & 12 & 32 & 49 & 75 & 97 \\ [2] & 4 & 10 & 11 & 25 & 48 \\ [3] & 5 & 7 & 21 & 31 & 98 \\ [4] & 1 & 9 & 17 & 34 & 46 \\ \hline \end{array}$$

Figure 2. Original matrix M (left) and row-by-row processed matrix (right).

Then you apply Binary Search to the first row. If you find Goldie, wonderful! You return its position (in the processed matrix, not in the original one). Otherwise, you proceed to the second row and so on. If you do not find Goldie after checking every row, you return the pair (-1,-1).

Proposal 3: Before visiting the elements of the matrix, you sort all the elements in such a way that if you read the matrix from top to bottom, from left to right, they are sorted from smallest to largest, as shown in Figure 3.

$$\begin{array}{c|ccccc|} & [0] & [1] & [2] & [3] & [4] \\ \hline [0] & 3 & 23 & 14 & 6 & 18 \\ [1] & 97 & 32 & 12 & 75 & 49 \\ [2] & 25 & 11 & 48 & 4 & 10 \\ [3] & 5 & 98 & 7 & 21 & 31 \\ [4] & 1 & 46 & 34 & 9 & 17 \\ \hline \end{array} \Rightarrow \begin{array}{c|ccccc|} & [0] & [1] & [2] & [3] & [4] \\ \hline [0] & 1 & 3 & 4 & 5 & 6 \\ [1] & 7 & 9 & 10 & 11 & 12 \\ [2] & 14 & 17 & 18 & 21 & 23 \\ [3] & 25 & 31 & 32 & 34 & 46 \\ [4] & 48 & 49 & 75 & 97 & 98 \\ \hline \end{array}$$

Figure 3. Original matrix M (left) and processed matrix (right)

Then you apply the following algorithm:

- Apply the principle of Binary Search to find the row where Goldie might be. That is, instead of checking every row (as in Proposal 2) you start checking the row in the middle of the matrix. If Goldie is not there, you decide whether to go to the row in the middle of the upper half of the matrix or the one in the middle of the lower half and so on. To know whether Goldie might be in the current row, just check if G is greater than or equal to A[row,0] (the leftmost element in the row) and less than or equal to A[row,N-1] (the rightmost element in the row). If so, Goldie might be there. If not, Goldie is definitely not in that row.
- Once you find a row where Goldie might be hiding, apply Binary Search to check whether Goldie is actually in that row. If found, return its position (in the processed matrix, not in the original one). If not, return (-1,-1)

2. YOUR TASK

Your mid-term task is divided in two parts:

Part 1: Analyse the 3 algorithmic proposals just described to then recommend one to be implemented. Your analysis must include:

a) The **pseudocode** description of each proposal. Your pseudocode might include tasks for which we have very well known algorithms, as binary search or sorting. In that case, the task must be encapsulated in a function. If the solution to the task is an algorithm we studied in the course, then you do not need to write the pseudocode for them, just call them. Notice that you need the version of Binary Search that returns the position of the number being searched for (instead of returning TRUE or FALSE). You can assume the following function definitions:

- Sort(A): receives as input argument a one dimensional array A and sorts the elements of that array from smallest to largest. In the case of sorting, you have several algorithms to choose from. Select the best algorithm for the task (the one combining simplicity and low time complexity). Justify your choice when finding an expression for the running time of the proposals requiring a sorting algorithm.

- Binary_Search(A,x): receives as input an array A and a number x and returns the position of the number in the array. If the number is not in the array, returns -1.
- Modified_Binary_Search(M,N,x): receives a **sorted** squared matrix of NxN elements and a number as input arguments. It returns the position (index) of the row where the number may be. Because the matrix is sorted, a number may be in a row if it is equal to or greater than the leftmost element in the row and equal to or less than the rightmost element in the row. If the number is not in any row, it returns -1.

b) The description of the **worst and best cases** for the execution time of each algorithm. Describe in detail the situation when the worst and best cases occur.

c) An **expression for the execution time, T(N),** for the worst and best case of each proposal. In deriving each expression, make sure you explain each step. Just writing the function describing the running time is not enough. You must show you know how to derive it from the pseudocode. Remember that here you need to justify your choice of a sorting algorithm for Proposals 2 and 3. Include at least one advantage and one drawback of your sorting choice. For proposals 2 and 3, it is useful to separate the contribution to the running time of the pre-processing of the matrix and the search for Goldie.

d) The **growth functions** of the execution time for each proposal, for worst and best case. For proposals 2 and 3, include a brief discussion of what part of the algorithm dominates the execution time: the pre-processing of the matrix or the search for Goldie.

e) The **Theta notation** for the worst and best cases of each proposal. In particular, find a set of values of c_1, c_2 and n_0 for which T(N) is Theta(g(N)). The values c_1, c_2 and n_0 can be actual numbers (if you derived T(N) counting up in detail every time unit) or algebraic (if you used generic constants for the expression of T(N)).

f) Based on the previous findings, discuss what proposal you would recommend to implement and why. This part is a **reflective writing exercise** where you consider different use cases (e.g. the matrix values do not change vs. the matrix values change every time you run the algorithm), worst and best cases and the reasons behind your recommendation. It is not enough to just name one proposal, you must discuss **why** this proposal is best. Maybe there is no a single proposal that is best for all cases.

You might want to check that your pseudocode and you final conclusion is correct by:

- Implementing a functional C++ version of each proposal. This will help you to check that your pseudocode is correct and your complexity analysis is correct (you can run each proposal for big matrix sizes and check that the fastest algorithm is also the one with the lowest time complexity).

- Comparing the growth function with the actual execution time of your code, for each proposal. To measure the execution time of your code, simply execute a variant of the time() function at the beginning and at the end of your code. This measurement is not exact and it has all the drawbacks discussed at the beginning of the course, but it allows you to make a quick comparative empirical analysis of the proposals. You can then plot the results: the horizontal axis is the number of rows (columns) of the matrix and the vertical axis is the execution time.

These two last activities (implementing the code and making the empirical comparison) are not compulsory and they will not be marked. That is, you do not need to report on this part of your work. You only need to report on parts a)-f). This suggestion is just given to you as a way to check the correctness of your work before submitting.
We encourage you to use C++, as this is the programing language used in the practical exercises of this module. However, as this part is not marked, you can use any programming language you want.

Part 2: Propose a new algorithmic approach for the problem of Where is Goldie? **using the idea of hashing**. Ideally (but this is not compulsory), there must be one situation where this new proposal has a lower time complexity that any of the 3 previous proposals. To describe your proposal you must include (with the same level of detail asked in the previous part of the assessment):

a) an **English description** of the idea behind the algorithm, in the same way proposals 1-3 were firstly explained. Use images if that facilitates the explanation.
b) a **pseudocode description** of the algorithm
c) The description of the **worst and best cases** of your proposal in terms of the running time.
d) An **expression for the running time, T(N),** of your proposal.
e) The growth function of the running time, for worst and best cases.
f) The **Theta notation** for the worst and best cases of each proposal.
g) A discussion on how your proposal is better (or not) than any of the proposals 1-3. Again, this is a **reflective writing exercise**.

As with Part 1, you can check the correctness of your work by implementing Proposal 4 on C++ (or any programming language you want to use).

3. DOCUMENTS

Your submission must contain a single PDF file addressing the points:

- a)-f) of Part 1
- a)-g) of Part 2

4. MARKING CRITERIA

This mid-term assessment is worth 40% of your final mark. This percentage is further divided as follows:

- Work on Proposal 1: 6%
- Work on Proposal 2: 6%
- Work on Proposal 3: 6%
- Reflective writing comparing proposals 1-3: 5%
- Work on proposal 4: 12%
- Reflective writing comparing proposal 4 vs. proposals 1-3: 5%

For proposals 1-3, we will mark your work according to the set of criteria shown below.

CRITERION	**Not addressed (0%)**	**Attempted, with major omissions (30%)**	**Attempted, with minor omissions (70%)**	**Complete work (100%)**	**Weight (out of 6%)**
Pseudocode					1.5%
Worst and best cases description					1%
T(N) (worst & best cases)					1%
Growth function					1%
Theta Notation (for worst & best cases)					1.5%

The reflective writing comparing proposals 1-3 will be marked as follows:

CRITERION	**Not addressed (0%)**	**Attempted, with major omissions (30%)**	**Attempted, with minor omissions (70%)**	**Complete work (100%)**	**Weight (out of 5%)**
One proposal is clearly identified as the "winner" one, based on its time complexity					2.5%
Comparison of 'winner' proposal to other proposals for different cases					2.5%

Proposal 4, which requires creativity from your side to device a new algorithm, will be marked according to the following criteria.

CRITERION	**Not addressed (0%)**	**Attempted, with major omissions (30%)**	**Attempted, with minor omissions (70%)**	**Complete work (100%)**	**Weight (out of 12%)**
Clear English description, uses hashing to solve the problem (or part of it)					6%
Pseudocode					1.5%
Worst and best cases description					1%
T(N) (worst & best cases)					1%
Growth function					1%
Theta Notation (for worst & best cases)					1.5%

The reflective writing comparing proposal 4 to previous ones will be marked as follows:

CRITERION	**Not addressed (0%)**	**Attempted, with major omissions (30%)**	**Attempted, with minor omissions (70%)**	**Complete work (100%)**	**Weight (out of 5%)**
Comparative description of Proposal 4 with respect to Proposal 1					1.25%
Comparative description of Proposal 4 with respect to Proposal 1					1.25%
Comparative description of Proposal 4 with respect to Proposal 1					1.25%
Conclusion regarding the goodness of Proposal 4					1.25%

Notice that, if you do not submit a PDF file, you risk your work not being marked.