**ALGORITHMS & DATA STRUCTURES**

**L. Ouarbya**

**TOPIC 1: ANALYSIS OF ALGORITHMS**

**WORKING SESSION 1**

In this working session you have to analyse one case where you are presented with a problem and several algorithmic solutions to that problem. You must analyse each solution proposal and then recommend the best one.

Learning Objectives:

1. Getting comfortable describing algorithms using pseudo-code
2. Gaining practice in calculating the growth function of algorithms
3. Gaining practice recommending the best algorithm to solve a problem

|  |
| --- |
| **CASE 1: WHERE IS WALLY?** |

|  |  |
| --- | --- |
|  | In this numerical version of Where is Wally? instead of finding a person in a page filled with dozens or hundreds of drawings of people doing amusing things (as shown in Figure 1), you must a find a number in a matrix of integers made of R rows and C columns.  No number is repeated in the matrix. The number representing Wally is entered as an input parameter (W) to the function FIND\_WALLY(M, R, C, W), where M is the matrix. |
| **Figure 1.** Where is Wally?  <https://www.independent.co.uk/arts-entertainment/books/news/wheres-the-brains-behind-wally-6261459.html> |  |

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

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

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



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 Wally, wonderful! You return his position (in the processed matrix). Otherwise, you proceed to the second row and so on. If you do not find Wally 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.



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 Wally might be. That is, instead of checking every row (as in Proposal 2) you start checking the row in the middle. If Wally is not there, you decide whether to go to the row in the middle of the upper half or the one in the middle of the lower half and so on. To know whether Wally is in the current row, just check that W is higher than or equal to A[row,0] (the first element in the row) and lower than or equal to A[row,C-1] (the last element in the row)
* Once you find a row where Wally might be hiding, apply Binary Search to that row to check whether Wally is actually in that row. If found, return its position. If not, return (-1,-1)

Your task during this working session is to analyse every proposal to determine the growth function of its running time and then recommend the best. To do so, you must:

1. **Describe every proposal using pseudo-code.** Make sure your pseudo-code is correct by using the example matrix shown above to test a few cases. At least, you should test 2 cases: one where Wally is in the matrix and one where Wally is not in the matrix. Assume you can call to the following functions:

* Sort(): receives as input argument an array and sorts the elements of that array from smallest to largest with a growth fucntion equal to NlogN (N: number of elements in the array).
* Binary\_Search(): receives as input an array and a number and returns the position of the number in the element. If the number is not in the array, returns -1.
* Modified\_Binary\_Search(): receives a matrix and a number as input arguments. It returns the position of the row where the number might be. In the number is not in any row, it returns -1.

Write your pseudo-code below. Add comments wherever necessary.

|  |
| --- |
| YOUR ANSWER:  function FIND\_WALLY1(M,R,C,W)  your pseudocode here…  for 0 =< i < r  for 0 =< j < c  if(M[i][j] == W)    return (R,C)  else  return (-1,-1)    function FIND\_WALLY2(M,R,C,W)  your pseudocode here…  for 0 =< i < r  sortedArray = bubblesort(M[i])  y = binary search(Sorted Array, W)  if( y != -1)  return i, y  }  return -1, -1    function FIND\_WALLY3(M,R,C,W)  your pseudocode here…  for 0 =< I < n  for 0 =< j < n |

1. Based on the pseudo-code you wrote in the previous part:
   1. Determine the growth function (**in terms of R and C**) of every proposal.
   2. What part of the algorithm determines the growth function of proposals 2 and 3? The processing of the matrix or the search for Wally?

|  |
| --- |
| YOUR ANSWER:  Part i)  Growth function of Proposal 1: n^2  Growth function of Proposal 2: n^2  Growth function of Proposal 3:  Part ii)  Proposal 2: The part of the algorithm that determines the growth function is…  Proposal 3: The part of the algorithm that determines the growth function is… |

1. Suppose that every time you have to find Wally, the matrix changes. In that case, what algorithm (proposal) would you recommend to use? Why?

|  |
| --- |
| YOUR ANSWER:  (\*) Recommended proposal:  (\*) Why: |

1. Suppose that we use always exactly the same matrix to find Wally, but the value to look for changes at each run. That is, we must process the matrix just once and then, we have to find a different value for Wally using the very same matrix over and over. In that case, the sensible thing to do is dividing your algorithm in two (for proposals 2 & 3):
   1. the function PROCESS\_M(M) processes the matrix and it is run only once (for proposals 2 & 3)
   2. the modified function FIND\_WALLY\_MOD(M,R,C,W) that receives the pre-processed matrix and immediately starts searching for Wally.

In this case, what version of FIND\_WALLY\_MOD() would you recommend to use? Why?

|  |
| --- |
| YOUR ANSWER:  (\*) Recommended proposal:  (\*) Why: |

**BONUS TRACK:** If you finish this part **before the end of this session**, show it to me. If it is ok, at the end of the year you will have two additional points added to your coursework mark. If you do not have enough time to do this exercise during the session, do it during your 7 hours of personal study. It is a good exercise to gain mastery in this topic.

* Implement **at least 2 proposals** using your favourite programming language. Ask the user to enter the number of rows (R) and columns (C) of the matrix. Set a counter to zero. Every time your programme checks an element of the matrix, increase the counter in one unit. Run your programme for (R,C)=(10,10), (100, 100), (1000, 1000) and (10000, 10000) and for a value of W that is not in the matrix.
* Make a graph with the value of the counter as a function of the values (R,C).
* Do these curves resemble the growth function you calculated for the proposals?