The Short and the Tall

R x C students sit in an matrix of R rows by C columns. The shortest in each row stands up and the tallest of these, A, remains standing. The tallest student in each column stands up and the shortest of these, B, remains standing.

If $A \neq B$, is $A > B$?

If A = B, (same height) we have a Saddle Point in the matrix

e.g. Matrix of Integers

				Min in Row	
	5	8	9	5	-- A
	6	7	2	2	
	7	1	9	1	
Max in Col	7	8	9		
	B				

In this case, not (A > B)

Saddle-Point

Given a RxC Matrix, A, i.e. R rows and C columns, a position (i,j) is a Saddle Point in a matrix, A, if the value at (i,j) (i.e. A(i,j)), is the minimum of its row and the maximum of its column, i.e.

Saddle_Pt((i,j), A) $\equiv$ A(i,j) is the minimum of row i and the maximum of col. j.

e.g.

A = 1 2 3
4 5 6
7 8 9

Position (2,0) is a Saddle Point with value 7
In a RxC matrix the rows are indexed from 0 to R-1 and the columns from 0 to C-1.
A(2,0) = 7 is minimum in row 2 and maximum in column 0.
There may be more than one position that is a Saddle Point,
e.g. an all zero matrix.

Problems

1. Find the positions of all Saddle Points.
2. Find the position of just one Saddle Point

To find the positions or locate the Saddle Points, if any, 'circle' all the minimums in the rows and 'square' all the maximums in the columns. The positions, if any, that are both 'circled' and 'squared' are the Saddle Points.

By analysing the properties of Saddle Points, a simpler algorithm can be divised.

Two Person Zero-Sum Game Payoff Matrix

A Zero-Sum (2-person) Game.

Each player is given a choice of strategies/moves and each player's loss it the other's gain.
In a payoff matrix, the rows are labelled by the 'row player' and the columns are labelled by the opposing player (column player) strategies/moves.
The entry at position (i,j) in the matrix, is the payoff to the Row player, in the event of the Row player playing strategy/move, i, and the column player playing stategy/move, j.

Example:
Paper(P), Rock (R), Scissors (S), Payoff Matrix

	P	R	S
P	0	1	-1
R	-1	0	1
S	1	-1	0

Minimax Criterion to choose a strategy/move.

The Minimax Criterion is one that minimises the maximum damage the opponent can do.

Principles of Game Theory

- Each player makes the best possible move.
- Each player knows that the opponent is also making the best possible move (for them).

Strictly Determined Game

A game is strictly determined if it has at least one Saddle Point.
In strictly determined games:

- All Saddle Points have the same (payoff) value
- Choosing the row and column through any Saddle Point gives the Minimax criterion for both players, i.e. the game is ‘solved’ by the use of this pure game criterion.
- The value of a strictly determined game is the value of the Saddle Point. A Fair Game has the value zero, otherwise the game is unfair or biased.

Theorem 1.

If positions (i,j) and (s,t) are Saddle Points then A(i,j) = A(s,t).

i.e. If Saddle_Pt((i,j), A) and Saddle_Pt((s,t), A)

then A(i,j) = A(s,t).

Proof:

A(i,j) = Min of row i

$\therefore$ (All k|0 ≤ k < C : A(i,j) ≤A(i,k))

In particular,

A(i,j) ≤A(i,t) -- A(i,j) is Min of row i

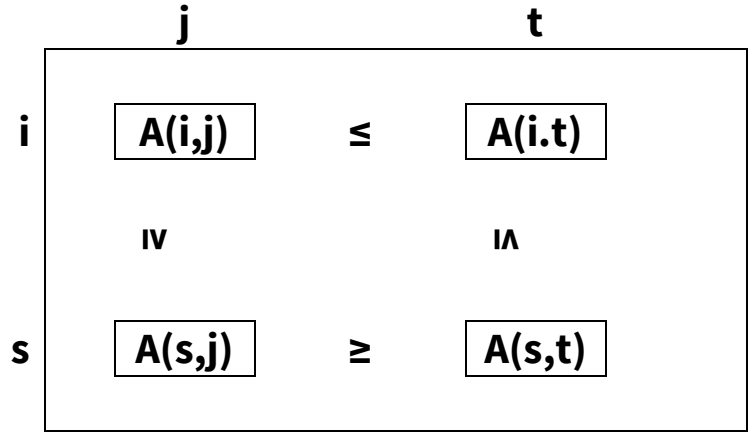
but A(i,t) ≤A(s,t) -- A(s,t) is Max of col t

$\therefore$ A(i,j) ≤A(s,t)

Similarly, A(s,t) ≤ A(i,j) as A(s,t) ≤ A(s,j) ≤ A(i,j)

$\therefore$ A(i,j) = A(s,t)

End Proof.



Theorem 2

Let

MinRow(i) = Minimum of Row i

MaxCol(j) = Maximum of Col j

then

$$\text{Saddle\_Pt}((i,j), A) \equiv \text{MinRow}(i) = \text{MaxCol}(j)$$

Proof:

($\Rightarrow$)

$$\text{Saddle\_Pt}((i,j), A) \Rightarrow A(i,j) = \text{MinRow}(i) \wedge A(i,j) = \text{MaxCol}(j)$$

tf.

$$\text{Saddle\_Pt}((i,j), A) \Rightarrow \text{MinRow}(i) = \text{MaxCol}(j)$$

($\Leftarrow$)

Assume MinRow(i) = MaxCol(j)

$\therefore$

MinRow(i) = A(i,y) some y s.t. $0 \leq y < C$

i.e. $(\exists\ y \mid 0 \leq y < C :$ MinRow(i) = A(i,y))

MaxCol(j) = A(x,j) some x s.t. $0 \leq x < R$

i.e. $(\exists\ x \mid 0 \leq x < R :$ MaxCol(j) = A(x,j))

Consider A(i,j)

$A(i,y) \leq A(i,j)$ -- A(i,y) = MinRow(i)

Also

$A(i,j) \leq A(x,j)$ -- A(x,j) = MaxCol(j)

$\therefore$

$A(i,y) \leq A(i,j) \leq A(x,j)$

From assumption MinRow(i) = MaxCol(j)

i.e $A(i,y) = A(x,j)$

i.e $A(i,y) = A(i,j) = A(x,j)$

$\therefore$ A(i,j) = MinRow(i)

and A(i,j) = MaxCol(j).

i.e. Saddle_Pt((i,j), A)

end Proof

Find All Saddle Points, if any

To find all Saddle Points. in A, find all (i,j)
s.t. MinRow(i) = MaxCol(j)

Find just one Saddle Point, if any

To find just one, we could start by finding all and exit having found the first.
But, we consider an alternative solution which will justify the claim in the 'The Short and the Tall" that if A=B then we have a Saddle Position.

Notation:

Let f be a function

$M = (\text{Max } k | 0 \leq k < n : f(k))$

iff $(\exists\, i | 0 \leq i < n \;\&\; M = f(i)) \;\wedge\; (\forall\, j |\, 0 \leq j < n : f(j) \leq M)$

Similarly for $(\text{Min } k | 0 \leq k < n : f(k))$

Theorem 3.

If Saddle_Pt((i,j),A) then

$A(i,j) = (\text{Max } k | 0 \leq k < n : \text{MinRow}(k))$

also

If Saddle_Pt((i,j),A) then

$A(i,j) = (\text{Min } k | 0 \leq k < n : \text{MaxCol}(k))$

Proof:

A(i,j) = MinRow(i) as Saddle_Pt((i,j),A)

also A(i,j) = MaxCol(j) as Saddle_Pt((i,j),A)

Show for all k s.t. $0 \le k < R$, MinRow(k) $\le$ A(i,j)

MinRow(k) $\le$ A(k,j) -- Min of row k,

$\le$ A(i,j) -- Max of col j

$\therefore$ A(i,j) = (Max k$|0 \le k < R$: MinRow(k))

Similarly,

A(i,j) = (Min k$|0 \le k < C$: MaxCol(k))

Theorem 4.

Let MinRow(mx) = (Max k$|0 \le k < R$: MinRow(k)) -- A in "Short & Tall"

Let MaxCol(mn) = (Min k$|0 \le k < C$: MaxCol(k)) -- B in "Short & Tall"

If MinRow(mx) = MaxCol(mn) then Saddle_Pt((mx,mn), A)

Proof:

From above:

Saddle_Pt((i,j), A) $\equiv$ MinRow(i) = MaxCol(j)

In particular,

if MinRow(mx) = MaxCol(mn)

then Saddle_Pt((mx,mn), A)

In Java,

```
public class Matrix_Test
{
    double[][] mat;
    Basic_Matrix bm;

    Matrix_Test()
    {
        String fname;

        TextIO.putln("Enter a file name: ");
        fname = TextIO.getWord();
        file2matrix(fname);
        TextIO.putln("\nThe input matrix is: ");
        print_matrix(mat);
        bm = new Basic_Matrix(mat);
        //one_saddle(bm);
        all_saddle(bm);
    } // Matrix_Test
```

```
void one_saddle(Basic_Matrix m)
{
    double[] minimum_rows;
    double[] maximum_cols;
    int index_maximum, index_minimum;
    Basic_Matrix mt;

    minimum_rows = m.min_row();
    mt = m.transpose();
    maximum_cols = mt.max_row();
    index_maximum = m.max_index(minimum_rows);
    index_minimum = m.min_index(maximum_cols);
    if (minimum_rows[index_maximum]==maximum_cols [index_minimum])
        print_saddle(m.mat, index_maximum, index_minimum);
    else
        TextIO.put("\nNo Saddle Point");
} // one_saddle
```

```
void all_saddle(Basic_Matrix m)
{
   double[] mins;
   double[] maxs;
   boolean found = false;
   Basic_Matrix mt;

   mins = m.min_row();
   mt = m.transpose();
   maxs = mt.max_row();
   for (int i = 0; i < m.rows; i = i+1)
        for (int j = 0;  j < m.cols; j = j+1)
           if ( mins[i] == maxs[j] )
           {
               found = true;
               print_saddle(m.mat,i,j);
           }
   if ( !found )
     TextIO.putln("\nNo Saddle Points");
} // all_saddle
```

```
void print_saddle( double[][] m, int i, int j)
{
    TextIO.put(m[i][j]);
    TextIO.put(" is a Saddle Point at ");
    TextIO.put("(" + i + ","+ j + ")" );
    TextIO.putln();
} // print_saddle

void file2matrix(String file_name)
{
    int rows, cols;

    TextIO.readFile(file_name);
    rows = TextIO.getInt();
    cols = TextIO.getInt();
    mat = new double[rows] [cols];
    for (int i = 0; i < rows; i = i+1)
        for (int j = 0;  j < cols; j = j+1)
            mat[i][j] = TextIO.getDouble();

} // file2matrix
```

```
    void print_matrix(double[][] mtx)
    {
        for (int i = 0; i < mtx.length; i = i+1)
        {
            print_array(mtx[i]);
        }
        TextIO.putln();
    } // print_matrix

    void print_array(double[] arr)
    {
          for (int k = 0;  k < arr.length; k = k+1)
                TextIO.put(arr[k] + " ");
          TextIO.putln();
    } // print_array

} // Matrix_Test
```

```
class Basic_Matrix
{
    int rows;            // #rows
    int cols;            // #columns
    double[][] mat;      // array of arrays

    Basic_Matrix(int m, int n)
    {// create m x n matrix of 0's
      rows = m;
      cols = n;
      mat = new double[m][n];
    } // Basic_Matrix

    Basic_Matrix(double[][] arr_2d)
    { // create matrix based on 2d array,
        rows = arr_2d.length;
        cols = arr_2d[0].length;
        mat = new double[rows][cols];
        for (int i = 0; i < rows; i=i+1)
            for (int j = 0; j < cols; j=j+1)
                    mat[i][j] = arr_2d[i][j];
    } // Basic_Matrix
```

```
Basic_Matrix transpose()
{
    Basic_Matrix result = new Basic_Matrix(cols, rows);
    for (int i = 0; i < cols; i=i+1)
        for (int j = 0; j < rows; j=j+1)
            result.mat[i][j] = mat[j][i];
    return result;
} // transpose

int min_index(double[] arr)
{
    int result = 0;
    double min = arr[0];
    for (int k=1; k < arr.length; k = k+1)
      if ( arr[k] < min )
      {
        min = arr[k];
        result = k;
      }
    return result;
} // min_index
```

```
double[] min_row()
{
    double[] result;  double [ ] row_k;
    result = new double [rows];
    for (int k=0; k < rows; k = k+1)
    {
         row_k = mat[k];
         result[k] = row_k[min_index(row_k)];
    }
     return result;
} // min_row

int max_index(double[] arr)
{
    int result = 0;
    double max = arr[0];
    for (int k=1; k < arr.length; k = k+1)
      if ( arr[k] > max )
      {
        max = arr[k];
        result = k;
      }
    return result;
} // min_index
```

```
     double[] max_row()
    {
        double[] result;
        double[] row_k;
        result = new double [rows];
        for (int k=0; k < rows; k = k+1)
        {
             row_k = mat[k];
             result[k] = row_k[max_index(row_k)];
        }
         return result;
    } // min_row

} //  Basic_Matrix
```