C:

```
#include <stdio.h>

#define MAX_H 300
#define MAX_W 300
#define TRUE 1
#define FALSE 0


int r,c,rsize,csize,global;
int m[MAX_H][MAX_W];
int rows[MAX_H],cols[MAX_W];
int temp[MAX_H];


int optimize_cols(){
	int ts=0,sum,current = 0;
	for(int j=0;j<c;j++){
		sum=0;
		for(int i=0;i<rsize;i++){
			sum += m[rows[i]][j];
		}
		if(sum>0){
			current += sum;
			temp[ts++] = j;
		}
	}
	if(current>global){
```

```
                global = current;
                for(int i=0;i<ts;i++){
                        cols[i] = temp[i];
                }
                csize = ts;
                return TRUE;
        }
        return FALSE;
}

int optimize_rows(){
        int ts=0,sum,current = 0;
        for(int i=0;i<r;i++){
                sum=0;
                for(int j=0;j<csize;j++){
                        sum += m[i][cols[j]];
                }
                if(sum>0){
                        current += sum;
                        temp[ts++] = i;
                }
        }
        if(current>global){
                global = current;
                for(int i=0;i<ts;i++){
                        rows[i] = temp[i];
                }
                rsize = ts;
                return TRUE;
```

```
        }
        return FALSE;
}

void solve(){
        rsize = csize = 1;
        int max_i = 0, max_j = 0;
        int max = m[0][0];
        for(int i=0;i<r;i++){
                for(int j=0;j<c;j++){
                        if(m[i][j]>max){
                                max_i = i;
                                max_j = j;
                                max = m[i][j];
                        }
                }
        }
        global = max;
        rows[0] = max_i;
        cols[0] = max_j;
        // printf("first: %d %d\n",max_i,max_j );
        while(TRUE){
                if(!optimize_rows()) break;
                //##################
                // printf("rows: ");
                // for(int i=0;i<rsize;i++) printf("%d ",rows[i] );
                // printf("\n");
                if(!optimize_cols()) break;
                //##################
```

```
                // printf("cols: ");
                // for(int i=0;i<csize;i++) printf("%d ",cols[i] );
                // printf("\n");
        }
        printf("%d %d\n",rsize,csize );
        for(int i=0;i<rsize;i++){
                printf("%d ",rows[i] );
        }
        printf("\n");
        for(int i=0;i<csize;i++){
                printf("%d ",cols[i] );
        }
        printf("\n");
}

int main(int argc, char const *argv[])
{
        scanf("%d%d",&r,&c);
        for(int i=0;i<r;i++){
                for(int j=0;j<c;j++){
                        scanf("%d",&m[i][j]);
                }
        }
        solve();
        return 0;
}
```

PYTHON:

```
import random
import pdb

h,w = map(lambda x : int(x), raw_input().split())
matrix = [None]*h
for i in range(h):
   matrix[i] = map(int,raw_input().split())

def score(rows, columns):
   total = 0
   for i in rows:
      for j in columns:
         total += matrix[i][j]
   return total

def crossover(male,female):
   rows = random.sample(male[0],random.randint(1,len(male[0])))
   rows += (random.sample(female[0],random.randint(1,len(female[0]))))
   rows = list(set(rows))
   columns = random.sample(male[1],random.randint(1,len(male[1])))
   columns += (random.sample(female[1],random.randint(1,len(female[1]))))
   columns = list(set(columns))
   # pdb.set_trace()
   row_sum = [0]*len(rows)
   for i in range(len(rows)):
```

```
        for j in range(len(columns)):
            row_sum[i] += matrix[rows[i]][columns[j]]
    rows = [x for _,x in sorted(zip(row_sum,rows),reverse=True)]
    rows = rows[0:len(rows)/2+1]

    column_sum = [0]*len(columns)
    # print(columns,rows)
    for i in range(len(columns)):
        for j in range(len(rows)):
            column_sum[i] += matrix[rows[j]][columns[i]]
    columns = [x for _,x in sorted(zip(column_sum,columns),reverse=True)]
    columns = columns[0:len(columns)/2+1]

    return [rows,columns,score(rows,columns)]

pcount = 10
generations = 25
population = []

for i in range(pcount):
    rows = random.sample(xrange(0,h),random.randint(1,h))
    columns = random.sample(xrange(0,w),random.randint(1,w))
    population.append([rows,columns,score(rows,columns)])

for g in range(generations):
    #crossover
    children = []
    for i in range(pcount*2):
        male,female = random.sample(population,2)
```

```
        children.append(crossover(male,female))


    new_population = population + children
    #selection
    population = sorted(new_population, key=lambda x : x[2],reverse=True)
    population = population[:pcount+1]
    # print(population[0][2])
# print("Final Solution" + str(population[0]))
# print(population[0])
print(str(len(population[0][0])) + " " + str(len(population[0][1])))
print(" ".join(map(str,sorted(population[0][0]))))
print(" ".join(map(str,sorted(population[0][1]))))
# print(population)
```