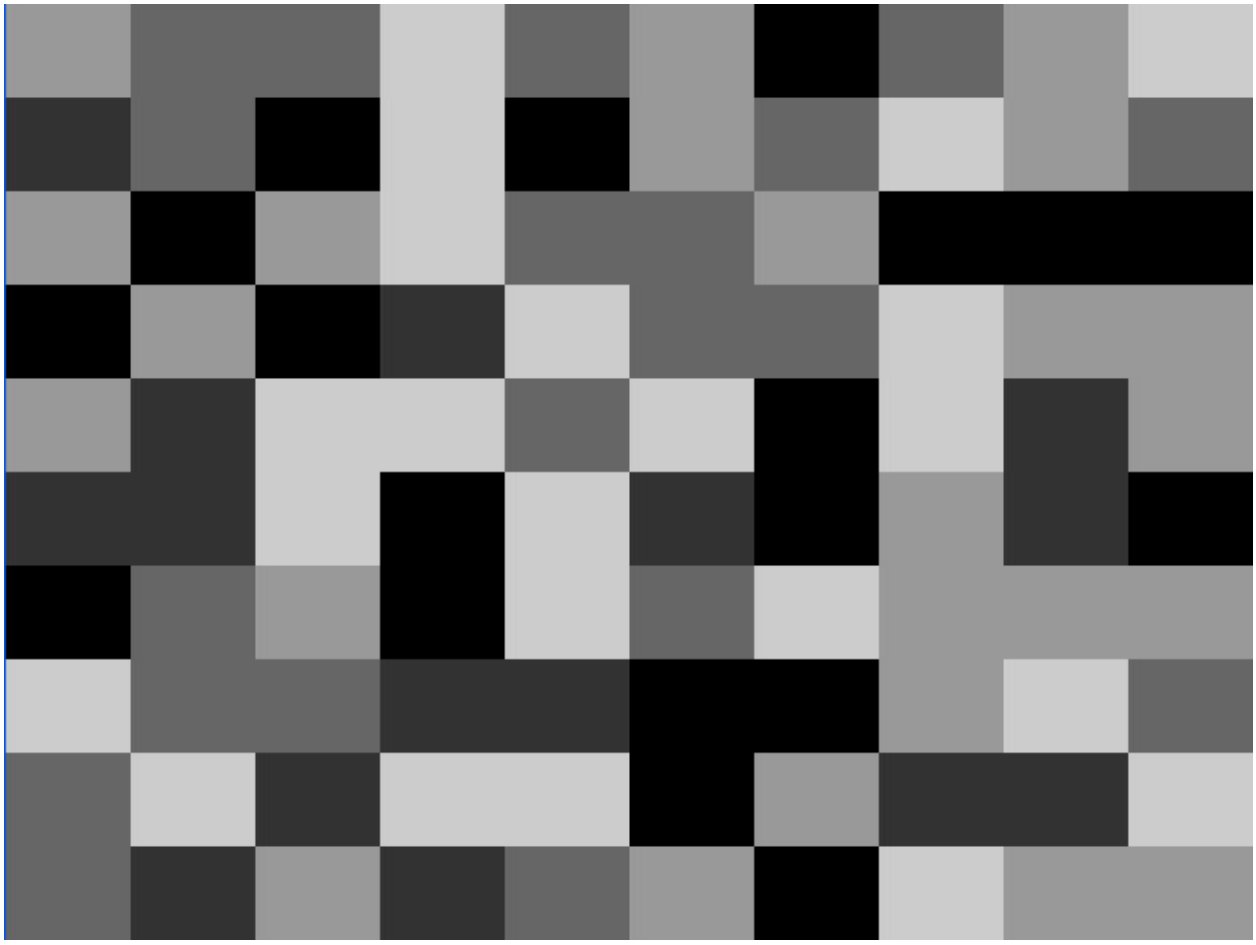
Ramsey Theory Search with Vectors



Ramsey theory, named after the British mathematician and philosopher Frank P. Ramsey, is a branch of mathematics that studies the conditions under which order must appear, often out of chaos. Ramsey theory questions are typically of the form: "how many elements of some structure must there be to guarantee that a particular property will hold?" In this project we ask the question, in an *m* by *m* array of *n* different characters, can we find a rectangular sub-array whose corners are all the same characters? Must such a sub-array exist, or is there an array that contains no such sub-array? For example, a 7 by 7 array of three characters may not contain such a sub-array, There are $3^{49} \sim 2.4e23$ such arrays to search through, and the search that produced the follow example involved about a million instances before finding this one:

```
3  1  2  1  2  3  2
3  1  3  3  3  1  2
2  3  3  1  2  3  1
1  3  1  1  3  2  2
1  1  2  2  3  3  1
2  3  2  3  1  2  1
2  1  1  2  1  3  2
```

To make things more interesting, the code below uses the Dark GDK library to display a checkerboard of colors, instead of numbers. The for-loop below is part of the `showMatrix()` function and uses the `dbRGB()` function to display shades of gray against a black (`dbRGB(0,0,0)`) background (you never see the background, but it needs to be there, I think.) Each number in the `mtrx[]` array produces a unique shade of gray ranging from `(255,255,255)` when `mtrx[]=1` to `(0,0,0)` when `mtrx[]=n`.

```
for(int j = 0; j < m; ++j)
{
       dbInk ( dbRGB(255*(n-mtrx[m*i+j])/(n-1),
                     255*(n-mtrx[m*i+j])/(n-1),
                     255*(n-mtrx[m*i+j])/(n-1)),dbRGB(0,0,0));
       dbBox(xinc*i,yinc*j,xinc*(i+1),yinc*(j+1));
}
```

The complete program is shown below, with some crucial parts missing. You job is to supply the missing parts.

In particular, the `searchMatrix()` function needs to specify proper indeces here:

```
if(matrix[i]==matrix[/*fill in index here*/] &&
   matrix[i]==matrix[/*fill in index here*/])
```

```
// Dark GDK - The Game Creators

/*Change the runtime library to /MT, from /MTd, for debug builds.
Project->(Project Name) Properties...->[+]Configuration Properties->[+]C/C++-
>Code Generation */

#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;

// whenever using Dark GDK you must ensure you include the header file
#include "DarkGDK.h"

// createMatrix creates a m by m matrix of random integers between 1 and n
void createMatrix(vector<int>& matrix, int m, int n)
{
	for(int i = 0; i < m*m; i++)
		matrix.push_back(1+rand()%n);
}

void showMatrix(vector<int>& mtrx, int m, int n)
{
	int xinc = 640/m, yinc = 480/m;
	for(int i = 0; i < m; i++)
	{
		for(int j = 0; j < m; ++j)
		{
			// Colors are set to shades of gray - experiment with this.
			dbInk ( dbRGB(255*(n-mtrx[m*i+j])/(n-1),
				255*(n-mtrx[m*i+j])/(n-1),
				255*(n-mtrx[m*i+j])/(n-1),dbRGB(0,0,0));
			dbBox(xinc*i,yinc*j,xinc*(i+1),yinc*(j+1));
		}
	}
}

bool searchMatrix(vector<int>& matrix, int m)
{
	//bool foundFlag = false;
	for(int i = 0; i < m*m-m-1; ++i)
		for(int j = i+1; j < (i/m+1)*m; ++j)
			if(i%m!=m-1 && matrix[i]==matrix[j])
				for(int k = i+m; k < m*m; k += m)
					if(matrix[i]==matrix[/*fill in index here*/] &&
					   matrix[i]==matrix[/*fill in index here*/])
					{
					/*cout << "\ni = " << i << endl
						<< "\nj = " << j << endl
						<< "\nk = " << k << endl
						<< "\nk+j-i = " << k+j-i << endl; */
					   return true;
					}
	return false;
}
```

```
// the main entry point for the application is this function
void DarkGDK ( void )
{
	// when starting a Dark GDK program it is useful to set global
	// application properties, we begin by turning the sync rate on,
	// this means we control when the screen is updated, we also set
	// the maximum rate to 60 which means the maximum frame rate will
	// be set at 60 frames per second
	dbSyncOn   ( );
	dbSyncRate ( 60 );

	// a call is made to this function so we can stop the GDK from
	// responding to the escape key, we can then add in some code in our
	// main loop so we can control what happens when the escape key is pressed
	dbDisableEscapeKey ( );

	vector<int> mtx;
	int m = 10, //the length of an edge of the square matrix
	    n = 5; //the number of different characters in the matrix
	createMatrix(mtx, m, n);
	showMatrix(mtx, m, n);
	if(searchMatrix(matrix, m))
	{
		dbText("Found a submatrix!");
	}
	// now we come to our main loop, we call LoopGDK so some internal
	// work can be carried out by the GDK
	while ( LoopGDK ( ) )
	{
		// empty function…might want to use it later for blinking
		// sub array?

		// here we make a call to update the contents of the screen
		dbSync ( );
	}

	// and now everything is ready to return back to Windows
	return;
}
```