Динамическое программирование — метод решения задачи путём её разбиения на несколько одинаковых подзадач, рекуррентно связанных между собой. Самым простым примером будут числа Фибоначчи — чтобы вычислить некоторое число в этой последовательности, нам нужно сперва вычислить третье число, сложив первые два, затем четвёртое таким же образом на основе второго и третьего, и так далее (да, мы слышали про замкнутую формулу).

Решение задачи динамическим программированием должно содержать следующее:

* Зависимость элементов динамики друг от друга. Такая зависимость может быть прямо дана в условии (так часто бывает, если это задача на числовые последовательности). В противном случае вы можете попытаться узнать какой-то известный числовой ряд (вроде тех же чисел Фибоначчи), вычислив первые несколько значений вручную
* Значение начальных состояний. В результате долгого разбиения на подзадачи вам необходимо свести функцию либо к уже известным значениям (как в случае с Фибоначчи — заранее определены первые два члена), либо к задаче, решаемой элементарно.

Алгоритм решения:

1. Считываем данные из входного файла.
2. Создаем двумерный массив boolean dp размером (M+1)x(N+1), где dp[i][j] будет равно true, если игрок, начинающий игру, может выиграть, начиная с клетки (i,j), и false в противном случае.
3. Заполняем dp[M][N] значением false, так как правая нижняя клетка отравлена.
4. Проходим по всем клеткам пирога в порядке слева-направо, снизу-вверх. Для каждой клетки (i,j) рассматриваем все возможные ходы: выбираем клетку (k,l), где k<i и l<j, и проверяем, может ли игрок выиграть, начиная с этой клетки. Если хотя бы один из таких ходов позволяет игроку выиграть, то dp[i][j] равно true, иначе - false.
5. Завершаем алгоритм. Все клетки, для которых dp[i][j] равно true, являются выигрышными ходами.

using System;

using System.IO;

using System.Collections.Generic;

namespace PoisonedCakeGame

{

class Program

{

static void Main(string[] args)

{

int m, n;

List<int> X = new List<int>();

// Чтение входных данных

using (StreamReader sr = new StreamReader("input.txt"))

{

string[] mn = sr.ReadLine().Split();

m = int.Parse(mn[0]);

n = int.Parse(mn[1]);

string[] x = sr.ReadToEnd().Split(new char[] { ' ', '\n', '\r' }, StringSplitOptions.RemoveEmptyEntries);

for (int i = 0; i < m; i++)

{

X.Add(int.Parse(x[i]));

}

}

// Определение выигрышных ходов

List<Tuple<int, int>> winningMoves = new List<Tuple<int, int>>();

bool[,] winningPositions = new bool[m + 1, n + 1];

for (int i = m - 1; i >= 0; i--)

{

for (int j = n - 1; j >= 0; j--)

{

if (!winningPositions[i, j])

{

winningMoves.Add(new Tuple<int, int>(i + 1, j + 1));

for (int k = 0; k < i; k++)

{

for (int l = 0; l < j; l++)

{

winningPositions[k, l] |= !winningPositions[i, j];

}

}

}

}

}

winningMoves.Reverse();

// Вывод результатов в файл

using (StreamWriter sw = new StreamWriter("output.txt"))

{

sw.WriteLine("для игры \"отравленный пирог\" " +

"используется прямоугольный пирог, разделенный на M \"строк\" " +

"горизонтальными разрезами и на N столбцов- вертикальными. т" +

"аким образом, пирог должен быть разбит на M на N клеток," +

" правая нижняя из которых \"отравлена\". Играют двое игроков, " +

"ходы делаются по очереди. Каждый ход заключается в том," +

" что игрок выбирает еще одну из несъеденных клеток пирога и съедает все клетки," +

" расположеные левее и выше выбранной (в том числе и выбранную). " +

"проигрывает тот кто съедает отравленную клетку.\r\n " +

"которая по заданной игровой позиции определяет " +

"все возможные выигрышные ходы для начинающего в этой позиции. " +

"\r\n\r\nданные во входном файле расположены в следующем порядке:" +

" M, N (1<=M, N<=9), X1, ..., Xm. З" +

"десь Xi - число оставшихся клеток в i-м снизу горизонтальном ряду." +

" Все числа во входном файле разделяются пробелами и/или символами перевода строки." +

"\r\nВыходные данные: в первую строку необходимо вывести количество " +

"различных выигрышных ходов K, а последующие K строк - сами выигрышные ходы. " +

"\r\nКаждый ход задается парой чисел (i,j), где i-номер (снизу) горизонтального ряда, " +

"а j-номер (справа) вертикального ряда, которому принадлежит выбранная клетка " +

"(1<=i<=m, 1<=j<=n)");

sw.WriteLine(winningMoves.Count);

foreach (var move in winningMoves)

{

sw.WriteLine(move.Item1 + " " + move.Item2);

}

}

}

}

}