# Numbers Game Puzzle

## The puzzle

Those familiar with Channel 4’s Countdown will be very familiar with this puzzle*. Given an unordered collection of integers as our initial state, can you reach a given target number through a series of simple mathematical operations?* Only the addition, subtraction, multiplication and division binary operators are permitted – furthermore, no subtractions are permitted to result in values <=0, and no divisions are permitted that would result in a non-zero remainder. Each time an operation is performed, two numbers are consumed, and one number, the result, is added, to our current numbers. To solve the puzzle, we need to generate the target number through a series of legal operations.

For example, given the starting numbers 10, 5 and 2, we can reach 52 [as 10\*5=50, then 50+2=52], or 30 [as 5-2=3, 10\*3=30] or 17 [10+5=15, 15+2=17] and so on.

## Numbers Game SDK

The puzzle is defined in the SDK in terms of the IGameSolver interface. This interface must be implemented by candidate algorithms – the algorithm developer’s primary task is to implement the GetSolution method of the IGameSolver interface. To do that, implementations are required to return a solution (modelled by ISolution), which describes all the steps that must be taken to solve the puzzle i.e. achieve the target value from the initial numbers. Solutions are essentially just an ordered list of operations, modelled by IOperation. The SDK usefully provides reusable Solution and Operation classes for use by algorithm developers, although they are not at all necessitated by the interface-based API. Alternative implementations of ISolution and IOperation are therefore supported.

The SDK also defines a Game namespace (NumbersGameSdk.Game), containing an entirely optional implementation of a model of a numbers game in progress, predictably named NumbersGame. Internally, this class models the current state of a game as an indexed list of integers, and maintains a history of operations as a stack of operations. Alternative implementations are certainly possible; this one is very simple. NumbersGame implements the INumbersGame interface, and can be used to ‘play’ a game, via sequential calls to DoOperation. (The Competition Runner actually uses NumbersGame internally to validate the solutions it receives from the algorithms being tested.)

NumbersGame provides methods specifically to assist solution-seeking algorithms: the interface provides an UndoOperation method, which enables algorithms to try out operations and then reverse those operations later on; it also provides two methods, CreateDescendent and CreateAllDescendents, which spawn entirely new game instances. The reference implementation DeepRecursiveUndoingSolver uses a single game instance and the DoOperation/UndoOperation methods as it looks for a solution. The reference ProgressiveRecursiveSolver uses multiple game instances generated using the CreateAllDescendents. New algorithm developers can reuse NumbersGame.

Alternatively, there are the CreateDescendent and CreateAllDescendents for generating entirely new Game instances from the current game (and some operation.)

looking for solutions. directly supports the application of an operation to a game via a DoOperation method, and can report its history as an enumerated collection of operations via its History property. In order to assist algorithm developers, it also has an Undo() method for reversing the last operation, and methods for spawning new Game instances

Reference Algorithms

The DeepRecursiveUndoingSolver reuses the Game object uses a single Game instance and the DoOperation/UndoOperation methods to arrive at its solution.

Numbers Game Competition Runner

Assemblies containing candidate algorithms must be located in the same directory as the runner, but there is otherwise no configuration required.

During initialisation, the SDK automatically examines the current executable’s folder location for assemblies that expose classes implementing IGameSolution. Any such classes are registered with the SDK’s SolverFactory, which also provides the means of enumerating through the registered classes.

The runner enumerates through all registered algorithms, and evaluates each one as follows:

The SDK defines a number of high-level interfaces, precisely describing the problem domain.

Candidate algorithms must implement the IGameSolver interface. IGameSolver contains the GetSolution method, defining that an array of integers, together with a target integer, shall be passed to your implementation, and that you are required to supply the ISolution out-parameter upon completion. Developers should return true to indicate a valid solution has been provided, and false to indicate no solution is possible.

The IGameSolver interface defines the GetSolution method as the means of executing compliant algorithms.

, which prescribes an array of input numbers, together with a target number, with an ISolution out-parameter.