RECURSION YEAH HOURS

Alexander De Baets

A quick note on the starter code and handout

Helpful Resources

- Recursion (Factorial, Binary Search)
- Recursion (Backtracking)
- Recursion (Fibonacci), Big O
- Remember to visit the CLaIR if you have conceptual questions! The CLaIR is Sunday-Thursday, 8PM-10PM on the second floor of Old Union.
- As always, the LaIR is there to help you work through your bugs! The LaIR is Sunday-Thursday 6PM-Midnight on the second floor of Old Union.

RECURSION

There are two parts to a recursive algorithm:

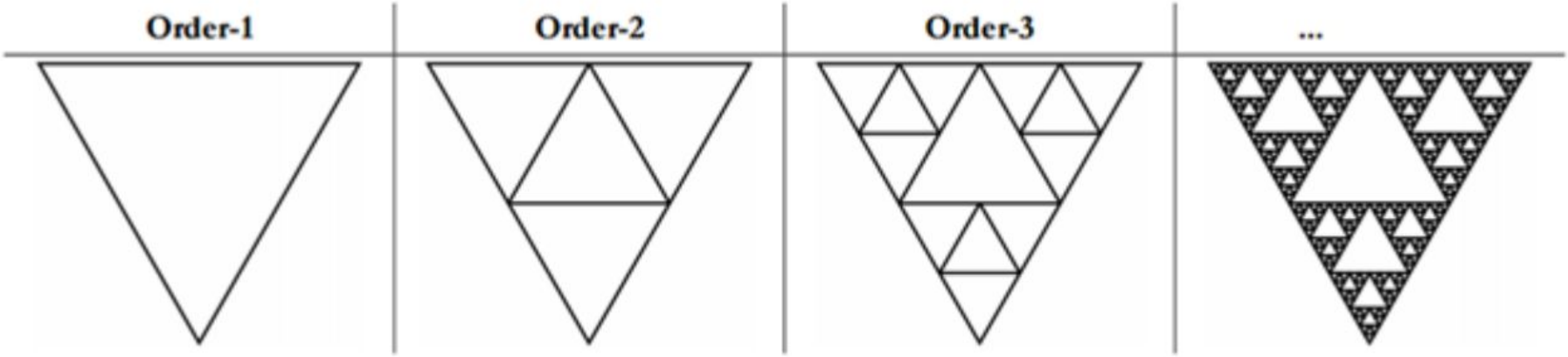
- The Base Case: The problem is so small we can easily solve it in a straightforward manner
- The Recursive Case: The problem is too big to be solved in a straightforward manner, so we break it into smaller subproblems and solve those.

PROJECT 1: HUMAN PYRAMID

Beware of how data is stored!

```
  col      0    1    2    3
row 0    {{A},
    1     {B, C},
    2     {D,  E,  F},
    3     {G,  H,  I,  J}}
```

PROJECT 2: SIERPINSKI TRIANGLE



Graphics? Yikes!

The Stanford `cslib` package

```
#include "gobjects.h"
```

`class GLine : public GObject`

This graphical object subclass represents a line segment. For example, the following code adds lines that mark the diagonals of the graphics window:

```
int main() {
   GWindow gw;
   cout << "This program draws the diagonals on the window." << endl;
   gw.add(new GLine(0, 0, gw.getWidth(), gw.getHeight()));
   gw.add(new GLine(0, gw.getHeight(), gw.getWidth(), 0));
   return 0;
}
```

Constructor

`GLine(x0, y0, x1, y1)`	Constructs a line segment from its endpoints.

Methods

`getEndPoint()`	Returns the point at which the line ends.
`getStartPoint()`	Returns the point at which the line starts.
`setEndPoint(x, y)`	Sets the end point in the line to (x, y), leaving the start point unchanged.
`setStartPoint(x, y)`	Sets the initial point in the line to (x, y), leaving the end point unchanged.

Things to beware of!

- When do you use Ints? When do you use Doubles?
- Are you drawing multiple lines in the same spot?
- Do not use a “pair” of functions which call each other

EXPLORATION VIA RECURSIVE BACKTRACKING

Base Case:

- We have found what we are looking for. We return the path that led us to this objective

Otherwise enter the Recursive Case:

- For every possible option (unless I've already faced this situation!)
  - "Choose" that option
  - Fully explore that option (Did I reach my objective?)
  - "Unchoose" that option

PROJECT 3: MARBLE SOLITAIRE

YOUR TASK

```
bool solvePuzzle(Grid<MarbleType>& board, Set<uint32_t>& exploredBoards,
                 Vector<Move>& moveHistory)
```

- `board` is the current game board configuration. More on what `MarbleType` is below, but the idea is that it keeps track of which spaces are currently occupied by a marble, which are free, and which are not playable (*i.e.* the four corners of the board where there are no marbles).
- `exploredBoards` is a set containing all the board configurations we have already explored. Because it is possible to reach a given board configuration via different sequences of moves, your recursive function should test if we have seen this `board` configuration before. If the current `board` is found in `exploredBoards`., return `false` to avoid repeating work. (Also be sure to add new boards to `exploredBoards`.) Note that the type is `Set<uint32_t>&`, not `Set<Grid<MarbleType>>&`, as you might expect! More on this below.
- `moveHistory` is the sequence of moves that led to the current board (not including human-played moves, if any). These are saved so that if/when a winning sequence is found and the function returns, the original calling function can reproduce the sequence of moves in the graphics display. More on what the `Move` type is below.

What you are given

In Marbles.cpp/h:

- makeMove
- undoMove
- findPossibleMoves

In compression.cpp/h:

- compressMarbleBoard

In marbletypes.ccp/h:

- Enum marble type

In marblegraphics.cpp/h:

- ALL THE GRAPHICS YAY!

Some words of advice

- DO NOT try to debug this with an entire board. It will drive you up a wall! Use the smaller boards we have given you.
- If your assignment is running slow, there are several things you should ask yourself:
  - Am I sending things through by reference or by value?
  - Do I search down the same path multiple times?

Good Luck!