(Provisional) Lecture 35: Game: Diagonals and Path Cleanup

10:00 AM, Nov 29, 2017

Contents

1 Diagonals

In class, Spike reviewed an implementation of retrieving the diagonals in a matrix.

Let's consider the following matrix:

$$\begin{matrix}1302\\5986\\4277\end{matrix}$$

First, let's review a recursive diagram for a function that takes as input a matrix, and returns the diagonals running from NE to SW in the input matrix:

- O.I. [[1; 3; 0; 2]; [5; 9; 8; 6]; [4; 1; 7; 7]]
  - R.I. [[5; 9; 8; 6]; [4; 1; 7; 7]]
  - R.O. [[]; [5]; [9; 4]; [8; 1]; [6; 7]; [7]]
- O.O [[1]; [3; 5]; [0; 9; 4]; [2; 8; 1]; [6; 7]; [7]]

To get from R.O. to O.O, we cons each element of the list that was peeled off - in this case, `[1; 3; 0; 2]` - onto the internal lists of R.O. That's why we add the empty list to the front of R.O. It makes creating the list containing just [1] much easier. However, there's still the leftover `[6; 7]` and [7], that are not consed on to.

The process of adding the elements of the peeled-off row to rest of the matrix is best handled by a helper. Let's create a recursive diagram for the helper, continuing with the same example.

- O.I. [[]; [5]; [9; 4]; [8; 1]; [6; 7]; [7]] and [1; 3; 0; 2]
  - R.I. [[5]; [9; 4]; [8; 1]; [6; 7]; [7]] and [3; 0; 2]
  - R.O. [[3; 5]; [0; 9; 4]; [2; 8; 1]; [6; 7]; [7]]
- O.O. [[1]; [3; 5]; [0; 9; 4]; [2; 8; 1]; [6; 7]; [7]]

For the helper, to go from the recursive output to the original output, you cons the head of the row to all of the internal rows of the matrix. So, you cons 1 to every inner list in the recursive output.

The base case for the helper is when the row is empty, in which case you return the rest of the matrix. In the example above, this would be when the input is [[6; 7]; [7]] and [], in which case the output would be [[6; 7]; [7]].

We started coding this together in class:

```
(*
diga_helper
input: an int list list, diags
       an int list, row
output: the result of consing each item in "row" onto the corresponding item
      in "diags," then copying any remaining items in diags.
*)
let rec diag_helper (diags: int list list) (row: int list) = match diags,
   row with
  | _, [] -> diags
  | hd1::tl1, hd::tl -> (hd::hd1):: (diag_helper tl1 tl) ;;

(*
input: an int matrix, mat
output: a list of int lists, each representing a NE to SW diagonal of mat
*)
let rec diagonals (rows: int list list) = match rows with
  | row::rest -> diag_helper([]::(diagonals rest)) row
  | [] -> failwith "matrix cannot be empty" ;;
```

These functions are not complete, though, and should be used as a stepping stone to get you started.

To get the other diagonals in the matrix - ones that go in other directions - we recommend flipping the matrix around and using the same function.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS17document by filling out the anonymous feedback form: http://cs.brown.edu/courses/cs017/feedback.