CS 325 Project 3 Report

Group 3

Date: November 20, 2014

Contributors: Cezary Wojcik, Sean McGlothlin, Matthew Eilertson

Recursive Function

We start with the following recursive function:

$$T[i,j] = \max \left\{ \begin{array}{l} \text{A[i,j]} \\ \text{T[i-1, j]} + \text{A[i, j]} \\ \text{T[i, j-1]} + \text{A[i, j]} \end{array} \right\}$$

The problem with this function is that it does not account for base cases. We can remedy this fairly simply:

$$T[i,j] = \max \left\{ \begin{array}{l} A[i,j] \\ \left\{ \begin{array}{ll} T[i-1,j] + A[i,j] & i > 0 \\ 0 + A[i,j] & i = 0 \end{array} \right. \\ \left\{ \begin{array}{ll} T[i,j-1] + A[i,j] & j > 0 \\ 0 + A[i,j] & j = 0 \end{array} \right. \end{array} \right\}$$

Now the formula will correctly find the best solution that ends at entry (i, j).

Pseudocode

```
HValue(H, i, j):
  if i < 0 or j < 0 then
    return 0
  else
    return H[i][j]
  end if
```

Running time: $\Theta(1)$

```
Heuristic(grid, i, j):
  H ← Array2D()
  for y = 0 to j do
    for x = 0 to i do
      H[x][y] ← grid[x][y] + max{ HValue(H, i-1, j), HValue(H, i, j-1), 0 }
    end for
  end for
  return H
```

Running time: $\Theta(i \times j)$

```
OptimalPath(grid)
    H ← Heuristic(grid, grid.rows - 1, grid.columns - 1)
    (x, y) ← max{elements in bottom row of H and last column of H}
    Path ← Array()
    Path.append((x, y))
    while x ≠ 0 and y ≠ 0 and not (H[x − 1][y] < 0 and H[x][y − 1] < 0) do
        if H[x − 1][y] > H[x][y − 1] and x > 0 then
            x ← x − 1
            Path.append((x, y))
        else
            y ← y − 1
            Path.append((x, y))
        end if
    end while
    return Path
```