1 Problem

By starting at the top of the triangle below and moving to adjacent numbers on the row below, the maximum total from top to bottom is 23.

3 7 4 2 4 6 8 5 9 3

That is, 3 + 7 + 4 + 9 = 23.

Find the maximum total from top to bottom of the triangle below:

75 95 64 17 47 82 18 35 87 10 20 04 82 47 65 19 01 23 75 03 34 88 02 77 73 07 63 67 99 65 04 28 06 16 70 92 41 41 26 56 83 40 80 70 33 41 48 72 33 47 32 37 16 94 29 53 71 44 65 25 43 91 52 97 51 14 70 11 33 28 77 73 17 78 39 68 17 57 91 71 52 38 17 14 91 43 58 50 27 29 48 63 66 04 68 89 53 67 30 73 16 69 87 40 31 04 62 98 27 23 09 70 98 73 93 38 53 60 04 23

NOTE: As there are only 16384 routes, it is possible to solve this problem by trying every route. However, Problem 67, is the same challenge with a triangle containing one-hundred rows; it cannot be solved by brute force, and requires a clever method! ;o)

2 Solution

```
import qualified Data.Map as Map
import Data.Maybe

data TriPath = TriPath
  { triPos :: [(Int, Int)]
  , triVals :: [Int]
  } deriving (Show, Ord, Eq)

vals =
  [75
  , 95, 64
  , 17, 47, 82
  , 18, 35, 87, 10
  , 20, 04, 82, 47, 65
  , 19, 01, 23, 75, 03, 34
  , 88, 02, 77, 73, 07, 63, 67
  , 99, 65, 04, 28, 06, 16, 70, 92
  , 41, 41, 26, 56, 83, 40, 80, 70, 33
  , 41, 48, 72, 33, 47, 32, 37, 16, 94, 29
  , 53, 71, 44, 65, 25, 43, 91, 52, 97, 51, 14
  , 70, 11, 33, 28, 77, 73, 17, 78, 39, 68, 17, 57
  , 91, 71, 52, 38, 17, 14, 91, 43, 58, 50, 27, 29, 48
  , 63, 66, 04, 68, 89, 53, 67, 30, 73, 16, 69, 87, 40, 31
  , 04, 62, 98, 27, 23, 09, 70, 98, 73, 93, 38, 53, 60, 04, 23]

positions :: Int → [(Int, Int)] → [(Int, Int)]
positions n xs
  | (length xs) ≡ n = xs
```

```
    | otherwise        = positions n (concat [xs, [(x', y')]])
      where (x, y) = last xs
            (x', y') = if x < y
              then (x + 1, y)
              else (1, y + 1)
triMaps = Map.fromList $ zip (positions 120 [(1, 1)]) vals
triMapLookup p = fromJust $ Map.lookup p triMaps
bestPathFrom :: (Int, Int) → TriPath
bestPathFrom (x, 15) = TriPath [(x, 15)] [triMapLookup (x, 15)]
bestPathFrom (x, y) = appendPathItem (x, y) bestPath'
  where path1 = bestPathFrom (x, y + 1)
        path2 = bestPathFrom (x + 1, y + 1)
        bestPath' = if (sum $ triVals path1) ⩾ (sum $ triVals path2)
                    then path1
                    else path2
appendPathItem :: (Int, Int) → TriPath → TriPath
appendPathItem (x, y) paths = TriPath pos' vals'
  where pos' = concat [(triPos paths), [(x, y)]]
        vals' = concat [(triVals paths), [triMapLookup (x, y)]]
main = do
  let bestFullPath = bestPathFrom (1, 1)
      bestPathLength = sum $ triVals bestFullPath
  putStrLn $ "The largest top-bottom path sum for the given tree is " ++ show bestPathLengt
```

3 Result

```
runhaskell problem18.lhs
The largest top-bottom path sum for the given tree is 1074.
```