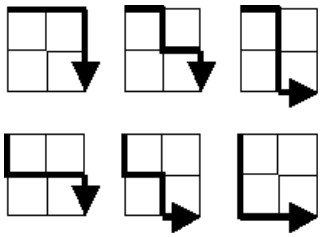
1 Problem

Starting in the top left corner of a 2 × 2 grid, there are 6 routes (without backtracking) to the bottom right corner.



How many routes are there through a 20 × 20 grid?

2 Method

1. Dynamic programming.
2. Memoization.
3. Data structure representation.

3 solution

```
import Data.Map as Map
import Control.Monad.State.Lazy as State

type StateMap a b = State (Map a b) b

  -- memoizeM' :: (Show a, Show b, Show c, Ord a, Ord b) =¿
  -- ((a a -¿ StateMap a b) -¿ (a -¿ StateMap a b)) -¿ (a -¿ b)
memoizeM' t x = evalState (f x) Map.empty where
  g x = do y ← t f x
     m ← get
     put $ Map.insert x y m
     newM ← get
     return y
  f x = get >>= λm → maybe (g x) return (Map.lookup x m)

wondM' :: Monad m ⇒ (Integer → m Integer) → Integer → m Integer
wondM' f' 1 = return 1
wondM' f' n = do
  let n' = if even n
           then (n `div` 2)
           else (3 * n + 1)
  n'' ← f' n'
```

```
  return (1 + n'')
wond' n = memoizeM' wondM' n
naiveRoutes :: Monad m => ((Int, Int) -> m Integer) -> (Int, Int) -> m Integer
naiveRoutes f' (x, y)
  | x == 20 && y == 20 - 1 = return 1
  | x == 20 - 1 && y == 20 = return 1
  | x == 20                = do
    y' <- f' (x, y + 1)
    return y'
  | y == 20                = do
    x' <- f' (x + 1, y)
    return x'
  | otherwise              = do
    x' <- f' (x + 1, y)
    y' <- f' (x, y + 1)
    return $ x' + y'
memRoutes (x, y) = memoizeM' naiveRoutes (x, y)
```

4 Result

```
[1 of 1] Compiling Main             ( src/euler/test.hs, interpreted )
Ok, modules loaded: Main.
*Main GOA Data.Map> memRoutes (0,0)
137846528820
it :: Integer
```

There are **137846528820** routes.