Article
Danny Wijnschenk · Nov 13, 2017 5m read

Advent of Code 2016 Day13: A Maze of Twisty Little Cubicles

This is a series of programming challenges for beginners and experienced Caché programmers.

For an introduction : go to article https://community.intersystems.com/post/advent-code-2016-day1-no-time-ta...

Today, you have to find a path through a maze. To know if a coordinate is a wall or an open space, you will have to do a calculation like this :

```
x*x + 3*x + 2*x*y + y + y*y
Add the office designer's favorite number (your puzzle input).
Find the binary representation of that sum; count the number of bits that are 1.
  - If the number of bits that are 1 is even, it's an open space.
  - If the number of bits that are 1 is odd, it's a wall.
```

The complete description is on http://adventofcode.com/2016/day/13, and it also includes your puzzle input.

There are two functions in Caché which will help you in the calculation that you might have never used before :

- $factor : to convert an integer to a $bit string (http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=...)
- $bitcount : to count the number of bits in a bitstring (http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=...)

The solution is pretty straightforward (especially if you survived the challenge on day 11) : try the possible moves (up, down, right, left) by calculating if you hit an open space or a wall, stopping if you reached a coordinate twice. You are finished when you reach location [31,39]

Here is my code :

```
Class AOC2016.Day13 Extends AOC2016.Utils
{

ClassMethod Part1()
{
  #Dim path as %String = ""
  #Dim posX as %Integer = 1
  #Dim posY as %Integer = 1
  Set %grid = ""
  Set %maxPath = 150
  Do ..Move(posX,posY,path)
}
```

```
ClassMethod Move(posX, posY, path) As %Boolean
{
  #Dim valid as %Boolean = 0
  #Dim deltaX, deltaY as %Integer
  #Dim move as %String
  If $ListLength(path)'<%maxPath Quit 0

  For move="0,1","1,0","0,-1","-1,0" {
    set deltaX=$P(move,",",1)
    set deltaY=$P(move,",",2)
    If $ListFind(path,$lb(posX+deltaX,posY+deltaY)) Continue
;been here already in this path
    If ..ValidMove(posX+deltaX,posY+deltaY) {
      If (posX+deltaX)=31,(posY+deltaY)=39 {
        Set valid = 1
        If $ListLength(path_$lb($lb(posX+deltaX,posY+deltaY)))<%maxPath {
          Set %maxPath = $ListLength(path_$lb($lb(posX+deltaX,posY+deltaY)))
          w !,%maxPath
        }
      } else {
        Set valid = ..Move(posX+deltaX,posY+deltaY, path_$lb($lb(posX+deltaX,posY+
deltaY)))
      }
    }
  }
  Quit valid
}

ClassMethod ValidMove(x, y) As %Boolean
{
  #Dim calc as %Integer
  #Dim valid as %Boolean = 1
  If (x<0)!(y<0) Quit 0
  If $Get(%grid(x,y))'="" Quit %grid(x,y)
  set calc=(x*x)+(3*x)+(2*x*y)+y+(y*y)+1358
  If $BitCount($Factor(calc),1)#2 set valid = 0
  Set %grid(x,y)=valid
  Quit valid
}

}
```

The second part of the challenge asks how many distinct locations you can reach in total in at most 50 steps, this will not have a big impact on the code for part 1.

Here is the code Bert wrote for this part of the challenge :

```
Start2() [History] PUBLIC {
  k History
  s input=1352
  for x=0:1:200 {
    f y=0:1:200 {
      s maze(x,y)= $$isWall(x,y,input)
```

```
    }
  }
  s options=1
  s options(1,"X")=1
  s options(1,"Y")=1
  s History("1,1")=""
  d doStep2(.maze,.options,1)
}

doStep2(Maze,Options,Step) [History] {
  s newOptions=0
  s nextOption=$ORDER(Options(""))
  while nextOption'="" {
    d calculateOptions(Options(nextOption,"X"),Options(nextOption,"Y"),.Maze,.
newOptions)
    s nextOption=$ORDER(Options(nextOption))
  }

  if Step<50 {
    d doStep2(.Maze,.newOptions,Step+1)
  } else {
    s count=0
    s visited=$ORDER(History(""))
    while visited'="" {
      s count=count+1
      s visited = $ORDER(History(visited))
    }
    w !,"total visited: ",count
  }
}

calculateOptions(X,Y,Maze,Options) [History] {
  if 'Maze(X+1,Y) && '$DATA(History((X+1)_","_Y)) {
    s Options=Options+1
    s Options(Options,"X")=X+1
    s Options(Options,"Y")=Y
    s History((X+1)_","_Y)=""
  }
  if 'Maze(X,Y+1) && '$DATA(History(X_","_(Y+1))) {
    s Options=Options+1
    s Options(Options,"X")=X
    s Options(Options,"Y")=Y+1
    s History(X_","_(Y+1))=""
  }
  if X>0 {
    if 'Maze(X-1,Y) && '$DATA(History((X-1)_","_Y)) {
      s Options=Options+1
      s Options(Options,"X")=X-1
      s Options(Options,"Y")=Y
      s History((X-1)_","_Y)=""
    }
  }
  if Y>0 {
    if 'Maze(X,Y-1) && '$DATA(History(X_","_(Y-1))) {
      s Options=Options+1
      s Options(Options,"X")=X
      s Options(Options,"Y")=Y-1
      s History(X_","_(Y-1))=""
    }
```

```
  }
}

isWall(X,Y,Input) {
  s result=0
  s number=(X*X) + (3*X) + (2*X*Y) + Y + (Y*Y)
  s number=number+Input
  s x=$FACTOR(number)
  s bits=$BITCOUNT(x,1)
  if (bits#2) {
    s result=1
  }
  q result
}
```

Look here for all our solutions so far : https://bitbucket.org/bertsarens/advent2016 and https://github.com/DannyWijnschenk/AdventOfCode2016

Here is the list of all Advent of Code 2016 articles :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

#Caché #Code Snippet #Contest #ObjectScript

Source URL:https://community.intersystems.com/post/advent-code-2016-day13-maze-twisty-little-cubicles