runWheresCroc(**makeMoves = myWC**, doPlot = T, showCroc = F, pause = 1, verbose = T, returnMem = F, mem = NA)

**myWC(moveInfo, readings, positions, edges, probs)**

Your function that takes five arguments:

1. **moveInfo**: A list of information for the move. This has two fields. The first is a vector of numbers called , where you will enter the moves you want to make. You should enter two moves (so you can move to a neighboring waterhole and search). Valid moves are the numbers of a neighboring or current waterhole or '0' which means you will search your current waterhole for Croc. The second field is a list called 'mem' that you can use to store information you want to remember from turn to turn.
2. **readings:** A vector giving the salinity, phosphate and nitrogen reading from Croc sensors at his current location.

readings[1] = **salinity**

readings[2] = **phosphate**

readings[3] = **nitrogen**

1. **positions** : A vector giving the positions of the two tourists (elements 1 and 2) and yourself (element 3). If a tourist has just been eaten by Croc that turn, the position will be multiplied by -1. If a tourist was eaten by Croc in a previous turn, then the position will be NA.

tourist1 = positions[1]

tourist2 = positions[2]

yourself = positions[3]

1. (4) **edges** : a two column matrix giving the edges paths between waterholes (edges) present (the numbers are from and to numbers for the waterholes). All edges can be crossed both ways, so are only given once.

|  |
| --- |
| getEdges=function() { |
|  | edges=matrix(c(1,2),ncol=2) |
|  | edges=rbind(edges,c(1,4)) |
|  | edges=rbind(edges,c(1,6))  …. |

1. **probs:** a list of three matrices giving the mean and standard deviation of readings for salinity, phosphate and nitrogen respectively at each waterhole. Your function should return the first argument passed with an updated moves vector and any changes to the 'mem' field you wish to access later on.

|  |
| --- |
| getProbs=function(){ |
|  | salinity=cbind(runif(40,100,200),runif(40,5,30)) |
|  | phosphate=cbind(runif(40,100,200),runif(40,5,30)) |
|  | nitrogen=cbind(runif(40,100,200),runif(40,5,30)) |
|  | list(salinity=salinity,phosphate=phosphate,nitrogen=nitrogen) |
|  | } |

## Given:

Random moving crocodile, sensor that gives 3 normal distributed conditions. **salinity, phosphate and nitrogen** levels. With a mean and sd.

Information about the waterholes. 3 distributions **salinity, phosphate and nitrogen.** With a mean and sd.

2 tourists: dead/alive and location.

Ranger: location and have/have not catch croc

## Goal:

Move the ranger to the crocodile.

## Solution:

Hidden markov model.

f(t) = f(t-1) \* transition \* observation (likelihood of all holes)

Observations: reading from crocodile (**salinity, phosphate and nitrogen**).

Transitions: edges between nodes.

Observation-matrix:

dim 40x40. For the three water properies. (salinity, phosphate and nitrogen). Calculate the probability density function (dnorm) between the water properties reading from the croc and the water properties data from the hole. Function arguments: readings and probs

salinity = dnorm(readings[1], probs$salinity[waterhole,1], probs$salinity[waterhole,2])

and the for the phosphate and nitrogen. Then multiply these three to get the probability. Save it in a matrix.

matrix[waterhole, waterhole] = salinity \* phosphate \* nitrogen

## Transition-matrix

Is fixed from beginning. dim 40x40. Function arguments: edges.

For every hole:

options = getOptions(waterhole, edges) # a list of neighbours of a waterhole

prob = 1/length(options) #the difficulty to move to the neighbours is the same

for (option in options) {

matrix[option, waterhole] = prob # save in matrix

}