Assignment2

R Markdown

```
rm(list = ls())

lapply(c('optrees', 'igraph'), require, character.only = TRUE)      #, 'qgraph'
```

```
## Loading required package: optrees
```

```
## Loading required package: igraph
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
```

Functions

```
#Matrix Representation
AdjMatrix2List <- function(d){
  ds <- which(!is.na(d), arr.ind = T)
  ds <- as.matrix(ds)
  temp <- matrix(0, nrow = nrow(ds), ncol = 1)

  for (i in 1:nrow(ds)) {
    temp[i] <- d[ds[i,1], ds[i, 2]]
  }

  ds <- cbind(ds, temp)
  ds <- ds[,c(2, 1, 3)]
  colnames(ds) <- c('Source', 'End', 'weight')
  return(ds)
}

#Minimum Spanning Tree
plot.mst <- function(arcList) {
 for (i in 1:nrow(arcList)){
   x0 <- x[arcList[i,1]]
   y0 <- y[arcList[i,1]]
   x1 <- x[arcList[i,2]]
```

```
   y1 <- y[arcList[i,2]]
   segments(x0, y0, x1, y1)
 }
}
```

Problems

Matrix Representation

```
n <- 1000
d <- runif(n*n)
d[d < 0.80] <- NA
d <- matrix(d, nrow = n, ncol = n)
diag(d) <- NA
d[upper.tri(d)] = t(d)[upper.tri(d)]    #upper.tri is upper triangle of a matrix and t is matrix transp
```

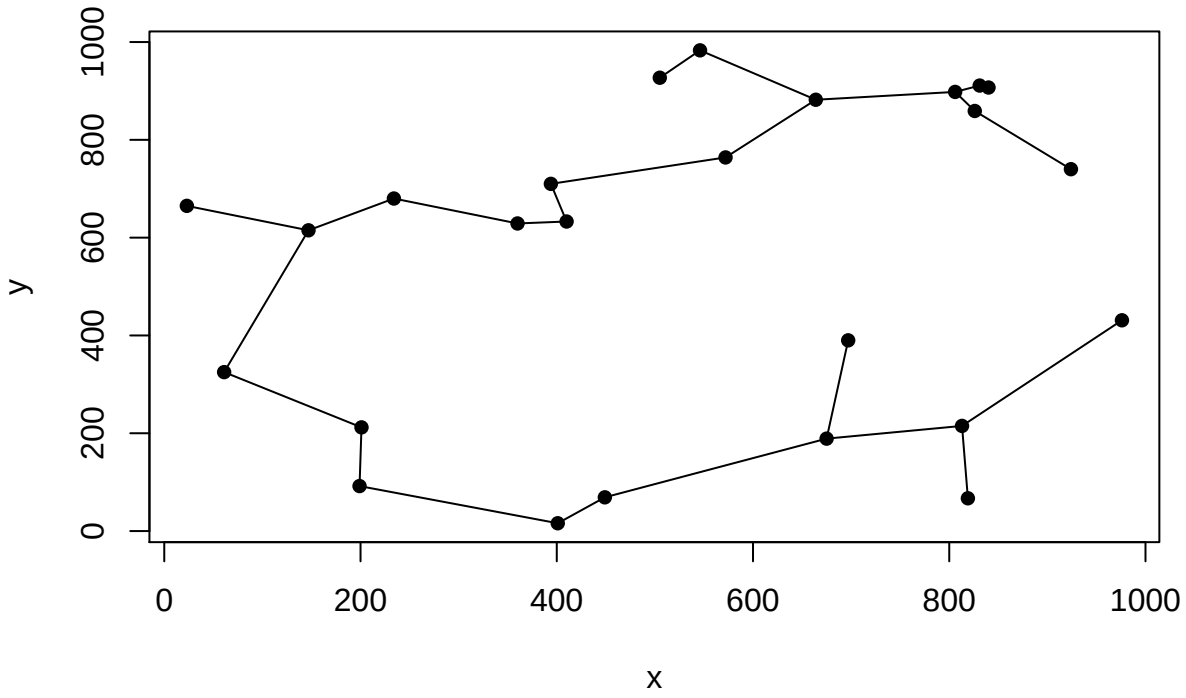
```
#Minimum Spanning Tree
n <- 25
x <- round(runif(n) * 1000)
y <- round(runif(n) * 1000)
plot(x, y, pch = 16)

d <- matrix(0, nrow = n, ncol = n)   #initialize sparse matrix
matrixy <- as.matrix(y)
matrixx <- as.matrix(x)

for (i in 1:length(x)) {
  matrow <- sqrt(((x[i] - matrixx)^2) + ((y[i] - matrixy) ^ 2))
  d[i,] <- matrow
}

ds <- AdjMatrix2List(d)

ds.mst <- msTreePrim(1:n, ds)
plot.mst(ds.mst$tree.arcs)
```



Hostile Agents

4.1: Minimum Spanning Tree Get from source node(start spy) to end node(receiving spy) while minimizing the cost(probability of getting caught) of doing so Any edge weights would be the probability of getting caught.

4.2: Inputs would be log(p(Gettin Caught)) or -log(1 - p(Getting Caught))

4.3: Kruskal's Algorithm as we are trying to minimize the probability of falling into hostile hands between each agent rather than the entire system as a whole.

4.4: Runtime: O(Edge lg Vertices)

Project Scheduling

Labels and nodes.

```
s.labels <- c('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j')
s.nodes <- c(90, 15, 5, 20, 21, 25, 14, 28, 30, 45)
```

Creating a matrix with estimated times from each node.

```
adjmat <- matrix(NA, nrow = length(s.labels), ncol <- length(s.labels))
dimnames(adjmat) <- list(s.labels, s.labels)
```

```
adjmat[2, 1] <- 90
adjmat[3, 2] <- 15
adjmat[4, 7] <- 14
adjmat[5, 4] <- 20
adjmat[6, 1] <- 90
adjmat[7, 3] <- 5
adjmat[7, 6] <- 25
adjmat[8, 4] <- 20
adjmat[9, 1] <- 90
adjmat[10, 4] <- 20
adjmat[10, 9] <- 30

adjlist <- AdjMatrix2List(adjmat * -1)
```

Bellman-Ford shortest path to get earliest start times and earliest finish times.

```
short <- getShortestPathTree(1:length(s.labels), adjlist, 'Bellman-Ford', show.data = FALSE, show.distar
dist <- matrix(0, nrow = length(s.labels), ncol = 1)
dist[,1] <- (short$distances * -1)
dist <- data.frame(dist)
date <- rep(as.Date('2017-11-1'), ncol(dist))
dist <- cbind(dist, date)
dist[,2] <- dist[,2] + dist[,1]
EF <- dist[,1] + s.nodes
dist[,3] <- EF
dist[,4] <- dist[,3] + dist[,2]
rownames(dist) <- s.labels
```

Bellman-Ford shortest path on the transpose of the graph to get latest finish times.

```
adjmatT <- matrix(NA, nrow = length(s.labels) + 1, ncol = length(s.labels) + 1)
t.labels <- c('dum' ,'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j')
t.nodes <- c(0, 90, 15, 5, 20, 21, 25, 14, 28, 30, 45)
dimnames(adjmatT) <- list(t.labels, t.labels)

adjmatT[11, 1] <- 0
adjmatT[9, 1] <- 0
adjmatT[6, 1] <- 0
adjmatT[5, 11] <- 45
adjmatT[10, 11] <- 45
adjmatT[2, 10] <- 30
adjmatT[5, 9] <- 28
adjmatT[4, 8] <- 14
adjmatT[7, 8] <- 14
adjmatT[2, 7] <- 25
adjmatT[5, 6] <- 21
adjmatT[8, 5] <- 20
adjmatT[3, 4] <- 5
adjmatT[2, 3] <- 15

adjlistT <- AdjMatrix2List(adjmatT * -1)

shortT <- getShortestPathTree(1:length(t.labels), adjlistT, 'Bellman-Ford', show.data = FALSE, show.dist
distT <- shortT$distances[-1]
dist[,5] <- distT * -1
```

Latest start and slack times.

```
LFdays <- ((distT * -1) - 194) * -1
dist[,5] <- LFdays
dist[,6] <- dist[,2] + LFdays
dist[,7] <- dist[,5] - s.nodes
dist[,8] <- dist[,2] + dist[,7]
dist[,9] <- LFdays - dist[,3]
colnames(dist) <- c(' ES Days','ES Date', 'EF Days', 'EF Date', 'LF Days', 'LF Date', 'LS Start', 'LS Da

finaldates <- dist[,c(2, 4, 6, 8)]
finaldays <- dist[,c(1, 3, 5, 7, 9)]
print(finaldates)
```

```
##      ES Date    EF Date    LF Date    LS Date
## a 2017-11-01 2018-01-30 2018-01-30 2017-11-01
## b 2018-01-30 2018-05-15 2018-05-20 2018-05-05
## c 2018-02-14 2018-06-04 2018-06-09 2018-06-04
## d 2018-03-10 2018-08-06 2018-08-06 2018-07-17
## e 2018-03-30 2018-09-16 2018-10-10 2018-09-19
## f 2018-01-30 2018-05-25 2018-05-25 2018-04-30
## g 2018-02-24 2018-07-03 2018-07-03 2018-06-19
## h 2018-03-30 2018-09-23 2018-10-10 2018-09-12
## i 2018-01-30 2018-05-30 2018-06-28 2018-05-29
## j 2018-03-30 2018-10-10 2018-10-10 2018-08-26
```

```
print(finaldays)
```

```
##    ES Days EF Days LF Days LS Start Slack
## a        0      90      90        0     0
## b       90     105     110       95     5
## c      105     110     115      110     5
## d      129     149     149      129     0
## e      149     170     194      173    24
## f       90     115     115       90     0
## g      115     129     129      115     0
## h      149     177     194      166    17
## i       90     120     149      119    29
## j      149     194     194      149     0
```

```
print(paste('Earliest completion date:', max(finaldates[,1])))
```

```
## [1] "Earliest completion date: 2018-03-30"
```

```
catdates <- paste(row.names(dist[dist$Slack > 0,]), collapse = ',')
catdays <- paste(row.names(dist[dist$Slack == 0,]), collapse = ',')
print(paste('Tasks with scheduling flexibility:', catdates))
```

```
## [1] "Tasks with scheduling flexibility: b,c,e,h,i"
```

```
print(paste('Tasks on critical path:', catdays))
```

```
## [1] "Tasks on critical path: a,d,f,g,j"
```