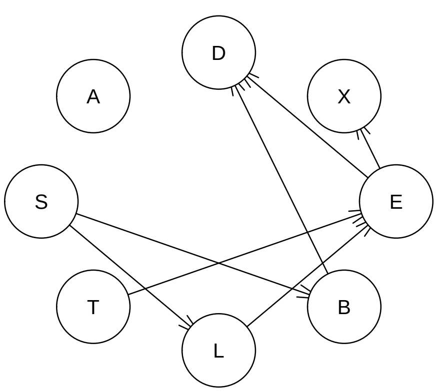
TDDE15 - Lab 1

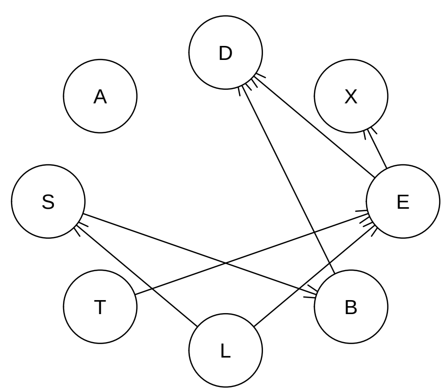
Question 1

```r
library(bnlearn)
data("asia")
structure = hc(asia, restart = 3)

b=T
for (i in 1:100) {
  structure2 = hc(asia, restart = 1)
  b = all.equal(structure, structure2)
  if (b!=TRUE) {
    print("Different network found")
    break
  }
}
```

```
## [1] "Different network found"
```

```r
plot(structure)
plot(structure2)
```



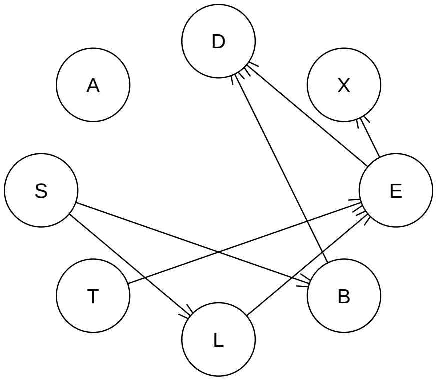


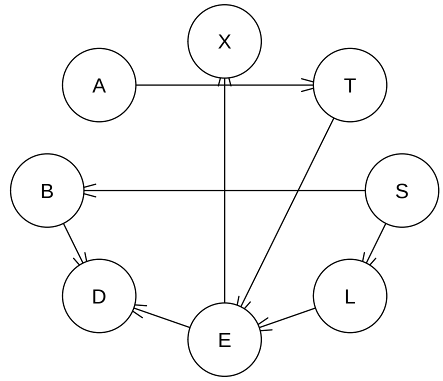
The HC return two different networks, because it can get trapped in a local optimum. In this case an edge is reversed, which would give the same score in the HC algorithm, therfore HC will evaluate these two networks as equal and not move between them.

Question 2

```r
library(gRain)
N = dim(asia)[1]

train = asia[1:floor(N*0.8),]
test = asia[(floor(N*0.8)+1):N,]

structure = hc(train, restart = 0)

# Learn conditional probabiliies given the nodes parents
fit = bn.fit(structure, data=train)

# Create Graphical independance network ( grain object )
fit_grain = as.grain(fit)

# create a junction tree and est. potential clique ( grain object )
# junc_tree = compile(fit_grain)
# print(junc_tree$cptlist)

# Remove S from test-data
test_ans = test[,"S"]
test_evid = subset(test, select = -2)

# Predict S
pred_s =c()
for (j in 1:dim(test_evid[1])) {

# Finding/evidance or potentials
# Need to extract observed values correctly.
  obs = c()
  for (i in 1:7) {
    obs = c(obs, as.character(test_evid[j,i]))
  }

  nodes_ev = names(test_evid)
  evid = setEvidence(fit_grain, nodes_ev, states = obs)

  # Querygrain to get conditional distributon
  node = c("S")
  prob_s = querygrain(evid, nodes = node)

  if (prob_s$S[1]>prob_s$S[2]) {
    pred_s=c(pred_s,"no")
  }else{
    pred_s=c(pred_s,"yes")
  }
}

misc_table = table(pred_s,test_ans)

# Correct DAG
dag = model2network("[A][S][T|A][L|S][B|S][D|B:E][E|T:L][X|E]")
```

```r
# Learn conditional probabiliies given the nodes parents
fit = bn.fit(dag, data=train)

# Create Graphical independance network ( grain object )
fit_grain = as.grain(fit)

# create a junction tree and est. potential clique ( grain object )
# junc_tree = compile(fit_grain)
# print(junc_tree$cptlist)

# Remove S from test-data
test_ans = test[,"S"]
test_evid = subset(test, select = -2)

#Predict S
pred_s =c()
for (j in 1:dim(test_evid[1])) {

# Finding/evidance or potentials
# Need to extract observed values correctly.
 obs = c()
  for (i in 1:7) {
    obs = c(obs, as.character(test_evid[j,i]))
  }

  nodes_ev = names(test_evid)
  evid = setEvidence(fit_grain, nodes_ev, states = obs)

  # Quergrain to get conditional distributon
  node = c("S")
  prob_s = querygrain(evid, nodes = node)

  if (prob_s$S[1]>prob_s$S[2]) {
    pred_s=c(pred_s,"no")
  }else{
    pred_s=c(pred_s,"yes")
  }
}

corr_table = table(pred_s,test_ans)

miss_class = function(conf_matr){
  return(1-sum(diag(conf_matr))/sum(conf_matr))
}

plot(structure)
plot(dag)
```






```
misc_table
```

```
##       test_ans
## pred_s  no yes
##    no  358 120
##    yes 147 375
```

```
miss_class(misc_table)
```

```
## [1] 0.267
```

```
corr_table
```

```
##       test_ans
## pred_s  no yes
##    no  358 120
##    yes 147 375
```

```
miss_class(corr_table)
```

```
## [1] 0.267
```

We get the same classification from the correct network. Thats because “S” is independend given the marcov blanket (“B” & “L”). The subgraph containing “S”, “B” and “L” is learned correctly.

Question 3

```r
marc_blanc = mb(fit, node = c("S" ))
marc_blanc
```

```
## [1] "B" "L"
```

```r
test_evid = subset(test_evid, select=marc_blanc)

pred_s =c()
for (j in 1:dim(test_evid[1])) {

# finding/evidance or potentials
#need to extract observed values correctly.....
  obs = c()
  for (i in 1:dim(test_evid)[2]) {
    obs = c(obs, as.character(test_evid[j,i]))
  }

  nodes_ev = names(test_evid)
  evid = setEvidence(fit_grain, nodes_ev, states = obs)
  #pEvidence(evid)

  # quergrain to get conditional distributon
  node = c("S")
  prob_s = querygrain(evid, nodes = node)

  if (prob_s$S[1]>prob_s$S[2]) {
    pred_s=c(pred_s,"no")
  }else{
    pred_s=c(pred_s,"yes")
  }
}

marcov_table = table(pred_s,test_ans)
misc_table
```

```
##       test_ans
## pred_s  no yes
##    no  358 120
##    yes 147 375
```

```r
miss_class(misc_table)
```

```
## [1] 0.267
```

```r
marcov_table
```

```
##       test_ans
## pred_s  no yes
##    no  358 120
##    yes 147 375
```

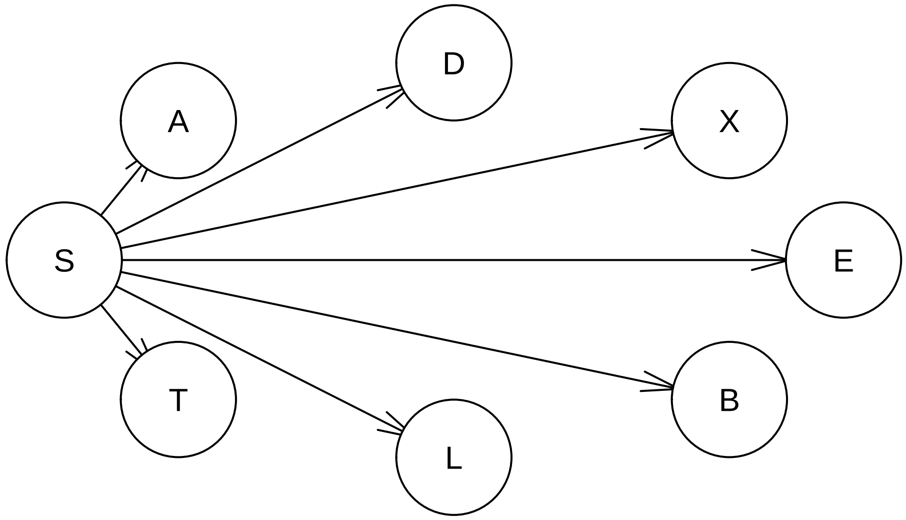
```r
miss_class(marcov_table)
```

```
## [1] 0.267
```

The classification turns out to be the same. Because “S” given the marcov blanket, is independent on the rest of the varibles given.

Question 4

```r
N = dim(asia)[1]
train = asia[1:floor(N*0.8),]
test = asia[(floor(N*0.8)+1):N,]
test_ans = test[,"S"]
test_evid = subset(test, select = -2)

#Crating an empty network
library(bnlearn)
b_net = empty.graph(names(asia))

# Adjacncy matrix (0L ensures that the number is stored as an integer instead of a double)
adj = matrix(0L, ncol = 8, nrow = 8,
             dimnames = list(names(asia), names(asia)))
amat(b_net) = adj

# Add edges in BN
for (i in names(test_evid)) {
  adj["S",i] = 1L
}
amat(b_net) = adj
plot(b_net)
```




```
bn_pot = bn.fit(b_net, data=train)
bn_grain = as.grain(bn_pot)

pred_s =c()
for (j in 1:dim(test_evid[1])) {
# finding/evidance or potentials
  obs = c()
  for (i in 1:dim(test_evid)[2]) {
    obs = c(obs, as.character(test_evid[j,i]))
  }

  nodes_ev = names(test_evid)
  evid = setEvidence(bn_grain, nodes_ev, states = obs)

  # quergrain to get conditional distributon
  node = c("S")
  prob_s = querygrain(evid, nodes = node)

  if (prob_s$S[1]>prob_s$S[2]) {
    pred_s=c(pred_s,"no")
  }else{
    pred_s=c(pred_s,"yes")
  }
}
naive_table = table(pred_s,test_ans)
```

```r
misc_table
```

```
##       test_ans
## pred_s  no yes
##    no  358 120
##    yes 147 375
```

```r
miss_class(misc_table)
```

```
## [1] 0.267
```

```r
naive_table
```

```
##       test_ans
## pred_s  no yes
##    no  389 180
##    yes 116 315
```

```r
miss_class(naive_table)
```

```
## [1] 0.296
```

Answer

The classification of the naive bayes classifer is predicting worse, with a higher missclassification rate, than the BN generated form the HC-algorithm. That is becaues Naive Bayes assumes that all varibles is independend given "S" and that there is a possible dependence between "S" and all other varibles. And in the correct DAG we saw that S is only dependent on "B" and "L".