```
class Caracter
  attr_accessor :caracter, :probabilidad, :codigo, :hijos
  def initialize(car="",prob=0)
    self.caracter = car
    self.probabilidad = prob
    self.codigo=""
    self.hijos=[]
  end
  def to_s
    "caracter:#{self.caracter}=>#{self.probabilidad}"
  end
  def calcular
    self.hijos.each do |hijo|
      self.probabilidad+=hijo.probabilidad
      self.caracter<<hijo.caracter
    end
  end
end
def recorrer(inicio,acumulado="")
  contador=0
  inicio.each do |item|
    item.codigo<<acumulado
    item.codigo<<contador.to_s
    if(item.hijos.size>0)
      recorrer(item.hijos,item.codigo)
    else
      DICC[item.caracter]=item.codigo
    end
    contador+=1
  end
end
print "cual es la cadena"
cadena = gets
DICC={}
caracteres = cadena.split(//)
unicos = caracteres.uniq
originales=[]
unicos.each do |i|
  originales<<Caracter.new(i,caracteres.count(i))
end
ordenado = originales.sort { |a,b| b.probabilidad <=> a.probabilidad  }
puts "ordenados"
puts ordenado
base = 2
while(ordenado.size>base)
  nuevo = Caracter.new
  base.times do
    nuevo.hijos<<ordenado.pop
  end
  nuevo.hijos.reverse!
  nuevo.calcular
  ordenado<<nuevo
  ordenado.sort! { |a,b| b.probabilidad <=> a.probabilidad  }
end
recorrer(ordenado)
p DICC.sort_by { |key,value| value.size }
codificado =""
caracteres.each do |i|
  codificado<<DICC[i]
end
puts codificado
decodificar = codificado.split(//)
aux = ""
recuperado = ""
dicc_inv = DICC.invert
p dicc_inv
decodificar.each do |car|
  aux<<car
```

```
  if(dicc_inv.has_key?(aux))
    #puts dicc_inv[aux]
    recuperado<<dicc_inv[aux]
    aux.replace("")
  end
end
puts recuperado
```