In [32]:

```
#3. Write a program to demonstrate the working of the decision tree
#based ID3 algorithm.Use an appropriate data set for building the
#decision tree and apply this knowledge to classify a new sample.


import math
import csv

def load_csv(filename):


    lines=csv.reader(open(filename,"r"));
    dataset = list(lines)
    headers = dataset.pop(0)
    #print(headers)
    #print(dataset)
    return dataset,headers
filename = "tennisdata.csv"
dataset = load_csv(filename)



class Node:
     def __init__(self,attribute):
        self.attribute=attribute
        self.children=[]
        self.answer=""




def subtables(data,col,delete):
            dic={}
            coldata=[row[col] for row in data]
            attr=list(set(coldata))

            counts=[0]*len(attr)
            r=len(data)
            c=len(data[0])
            for x in range(len(attr)):
                 for y in range(r):
                        if data[y][col]==attr[x]:
                              counts[x]+=1
            for x in range(len(attr)):
                dic[attr[x]]=[[0 for i in range(c)] for j in range(counts[x])]
                pos=0
                for y in range(r):
                    if data[y][col]==attr[x]:
                        if delete:
                            del data[y][col]

                        dic[attr[x]][pos]=data[y]
                        pos+=1

            return attr,dic



```

```
def entropy(S):
    attr=list(set(S))
    if len(attr)==1: #if all are +ve/-ve then entropy =0
        return 0
    counts=[0,0] # only two values possible 'yes' or 'no'
    for i in range(2):
        counts[i]=sum([1 for x in S if attr[i]==x])/(len(S)*1.0)

        sums=0
    for cnt in counts:
        sums+=-1*cnt*math.log(cnt,2)

    return sums



def compute_gain(data,col):
        attr,dic = subtables(data,col,delete=False)

        total_size=len(data)
        entropies=[0]*len(attr)
        ratio=[0]*len(attr)

        total_entropy=entropy([row[-1] for row in data])
        for x in range(len(attr)):
            ratio[x]=len(dic[attr[x]])/(total_size*1.0)
            entropies[x]=entropy([row[-1] for row in dic[attr[x]]])
            total_entropy-=ratio[x]*entropies[x]
        return total_entropy



def build_tree(data,features):
        lastcol=[row[-1] for row in data]



        if(len(set(lastcol)))==1: #if all are YES OR if all are no
            node=Node("")
            node.answer=lastcol[0]
            return node
        n=len(data[0])-1   #5-1=4
        gains=[0]*n  #'gains ',[0,0,0,0]
        for col in range(n):
            gains[col]=compute_gain(data,col)
        split=gains.index(max(gains))
        node=Node(features[split])
        fea = features[:split]+features[split+1:]

        attr,dic=subtables(data,split,delete=True)

        for x in range(len(attr)):
                child=build_tree(dic[attr[x]],fea)
                node.children.append((attr[x],child))
        return node
```

```
121
122 def print_tree(node,level):
123     if node.answer!="":
124         print(" "*level,node.answer)
125         return
126     print(" "*level,node.attribute)
127     for value,n in node.children:
128
129         print(" "*(level+1),value)
130         print_tree(n,level+2)
131
132
133
134
135 '''Main program'''
136
137 dataset,features=load_csv("tennisdata.csv")
138 #print(dataset)
139
140
141
142 # print(features)
143
144 node=build_tree(dataset,features)
145
146 print("The decision tree for the dataset using ID3 algorithm is")
147 print_tree(node,0)
148
149
```

```
The decision tree for the dataset using ID3 algorithm is
 Outlook
  Sunny
   Humidity
    Normal
     Yes
    High
     No
  Overcast
   Yes
  Rainy
   Windy
    True
     No
    False
     Yes
```

In [ ]:

```
1
```