In [1]:

```
from IPython.display import Markdown, display
def printmd(string):
    display(Markdown(string))
```

In [36]:

```
printmd('## **MT19017- Shubham Verma**')
```

MT19017- Shubham Verma

In [2]:

```
printmd('## **Hyperclique Pattern Discovery**')
```

Hyperclique Pattern Discovery

In [3]:

```
printmd('### **1. size 1 prevalent item**')
```

1. size 1 prevalent item

In [4]:

```
def f1_item():
    pre_it=[item for elem in data for item in elem]
    pre_it=list(set(pre_it))

    pre_items=[]
    for item in pre_it:
        pre_items.append([item])

    #convert item set into frozenset
    pre_items=list(map(frozenset,pre_items))
    return pre_items
```

In [5]:

```
printmd('### **2. support based prunning**')
```

2. support based prunning

In [6]:

```
def support_prunning(pattern, min_sup):
    dic={}
    n=len(data)
    for d in data:
        for p in pattern:
            if p.issubset(d):
                if not p in dic:
                    dic[p]=1
                else:
                    dic[p] += 1

    support=[]
    item=[]
    for d in dic:
        dic[d]/=n
        if dic[d] >= min_sup:
            support.append(dic[d])
            item.append(d)

    return item,support
```

In [7]:

```
printmd('### **3. H-Confidence based prunning**')
```

3. H-Confidence based prunning

In [8]:

```
def hc_prunning(item_set, sup, min_hc, item1, sup1):
    hconf=[]
    new_item=[]
    new_sup=[]
    for it,sp in zip(item_set, sup):
        item=list(it)
        max_sup=0
        for i in item:
            temp=sup1[item1.index(i)]
            if max_sup<temp:
                max_sup=temp

        hc= sp/max_sup
        if hc > min_hc:
            hconf.append(hc)
            new_item.append(it)
            new_sup.append(sp)

    return new_item,new_sup,hconf
```

In [9]:

```
printmd('### **4. Generate new item set F(k) using F(k-1)**')
```

4. Generate new item set F(k) using F(k-1)

In [10]:

```
def aprioriGen(pattern, size):
    new_p=[]
    n=len(pattern)
    size-=1
    for i in range(n):
        for j in range(i+1,n):
            p1=list(pattern[i])
            p2= list(pattern[j])

            list1=p1[:size-1]
            list2=p2[:size-1]
            list1.sort()
            list2.sort()
            if list1==list2:
                new_p.append(pattern[i] | pattern[j])
    return new_p
```

In [11]:

```
printmd('### **5. Hyperclique miner algorithm**')
```

5. Hyperclique miner algorithm

In [12]:

```
def Hyperclique_miner(min_hc, min_sp):
    pre_items = f1_item()                                   #find size 1 prevalent
 items
    n = len(pre_items)
    patterns = []
    support = []
    hconf = []
    item, sup = support_prunning(pre_items, min_sp)      #prunning on the basis
 of support
    zipped_pairs = zip(sup, item)                           #sort items on the basi
s of support
    item = [j for i,j in sorted(zipped_pairs)]
    sup1=sorted(sup)

    item1=[]
    for f_set in item:                                      #frozenset to list
        item1.append(list(f_set)[0])
    k=2
    new_list_length=len(item)
    size=0
    while(k<n and new_list_length>0 ):
        new_p = aprioriGen(item,k)                          # F(k-1) to F(k)
        item, sup = support_prunning(new_p, min_sp)      #prunning on the basis
 of support
        item, sup, hc = hc_prunning(item, sup, min_hc, item1, sup1) #prunning on
the basis of H-confidence
        new_list_length=len(item)
        size+=new_list_length
        if(new_list_length>0):
            patterns.append(item)
            support.append(sup)
            hconf.append(hc)
            k+=1

    return patterns,support,hconf,size
```

In [13]:

```
printmd('### **6. Read dataset using file handling**')
```

6. Read dataset using file handling

In [14]:

```
dataset=[line.rstrip('\n').split(" ") for line in open("Dataset/kosarak.dat")]
for i in range(len(dataset)):
    dataset[i]=[int(i) for i in dataset[i]]

len(dataset)
```

Out[14]:

990002

In [15]:

```
%matplotlib inline
from matplotlib import pyplot as plt
from time import time
from random import sample
```

In [16]:

```
printmd('### **7. Sample output of first 1000 entries**')
```

7. Sample output of first 1000 entries

In [17]:

```
data=dataset[:1000]
```

In [18]:

```
patterns,sup,hcon,total_size= Hyperclique_miner(0.2,0.01)
```

In [19]:

```
patterns
```

Out[19]:

```
[[frozenset({1, 3}),
  frozenset({6, 11}),
  frozenset({7, 27}),
  frozenset({3, 11}),
  frozenset({3, 6}),
  frozenset({1, 11}),
  frozenset({1, 6}),
  frozenset({85, 86}),
  frozenset({83, 85}),
  frozenset({83, 86}),
  frozenset({7, 83}),
  frozenset({148, 218}),
  frozenset({27, 83})],
 [frozenset({3, 6, 11}), frozenset({83, 85, 86})]]
```

In [20]:

```
patterns[0] #size 2 patterns
```

Out[20]:

```
[frozenset({1, 3}),
 frozenset({6, 11}),
 frozenset({7, 27}),
 frozenset({3, 11}),
 frozenset({3, 6}),
 frozenset({1, 11}),
 frozenset({1, 6}),
 frozenset({85, 86}),
 frozenset({83, 85}),
 frozenset({83, 86}),
 frozenset({7, 83}),
 frozenset({148, 218}),
 frozenset({27, 83})]
```

In [21]:

```
patterns[1] #size 3 patterns
```

Out[21]:

```
[frozenset({3, 6, 11}), frozenset({83, 85, 86})]
```

In [22]:

```
#support of items
sup
```

Out[22]:

```
[[0.094,
  0.347,
  0.044,
  0.166,
  0.265,
  0.093,
  0.145,
  0.015,
  0.013,
  0.013,
  0.02,
  0.055,
  0.015],
 [0.147, 0.013]]
```

In [23]:

```
# H-confidence
hcon
```

Out[23]:

```
[[0.20568927789934355,
  0.580267558528428,
  0.4444444444444444,
  0.36323851203501095,
  0.44314381270903014,
  0.23969072164948452,
  0.24247491638795987,
  0.9375,
  0.3939393939393939,
  0.3939393939393939,
  0.20202020202020202,
  0.7534246575342466,
  0.21739130434782605],
 [0.24581939799331104, 0.3939393939393939]]
```

In [24]:

```
# Total number of patterns
total_size
```

Out[24]:

```
15
```

In [25]:

```
printmd('### **9. Result and analysis**')
```

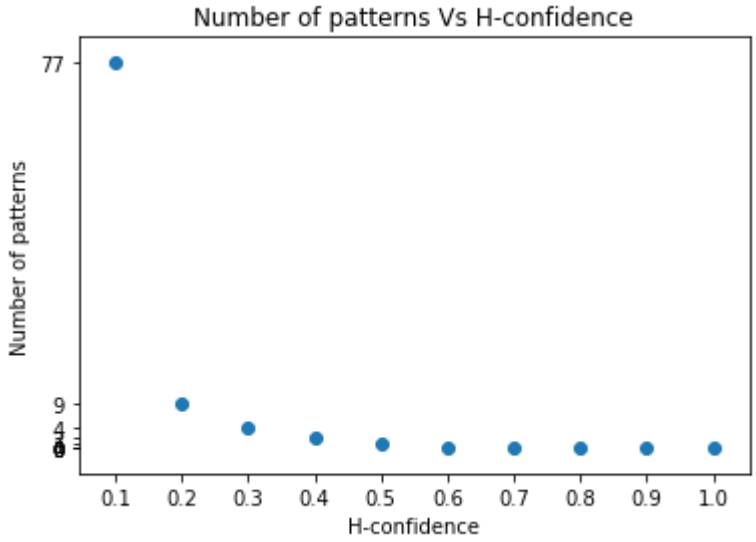
9. Result and analysis

In [28]:

```
hconf=[]
support=[]
length=[]
clock=[]
s=0.01
h=0.1
for i in range(10):
    t1=time()
    patterns,sp,hc,size= Hyperclique_miner(h,s)
    t2=time()-t1
    clock.append(t2)
    hconf.append(h)
    support.append(s)
    length.append(size)
    h+=0.1
    s+=0.02
```

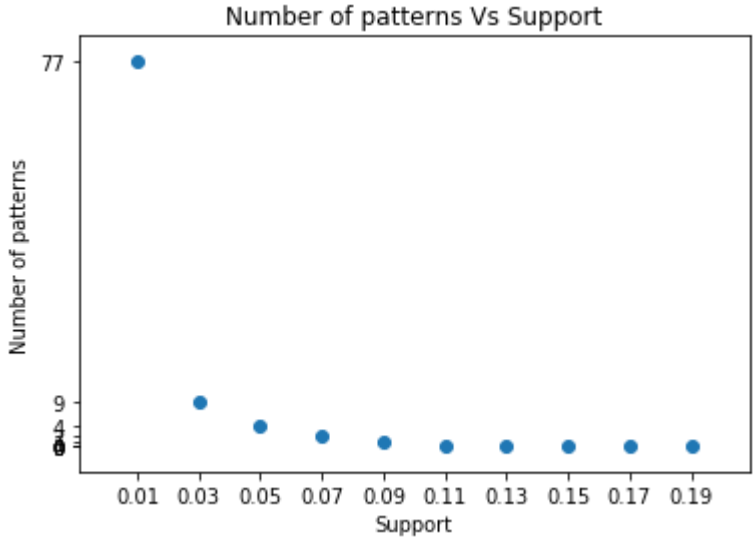
In [29]:

```
plt.xticks(hconf)
plt.yticks(length)
plt.xlabel('H-confidence')
plt.ylabel('Number of patterns')
plt.title('Number of patterns Vs H-confidence')
plt.scatter(hconf,length)
plt.show()
```



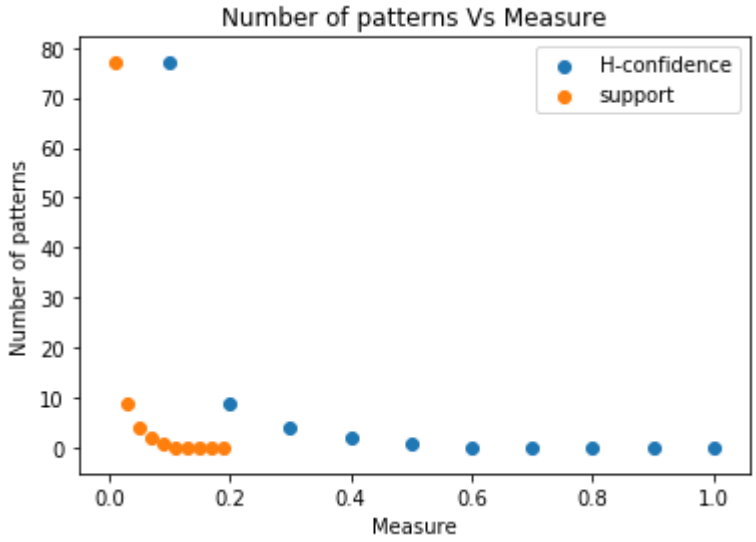
In [30]:

```
plt.xticks(support)
plt.yticks(length)
plt.xlabel('Support')
plt.ylabel('Number of patterns')
plt.title('Number of patterns Vs Support')
plt.scatter(support,length)
plt.show()
```



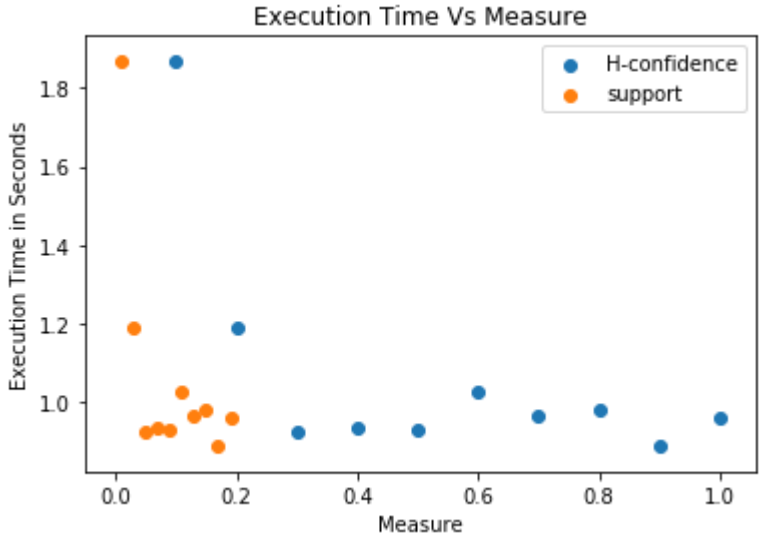
In [31]:

```
plt.xlabel('Measure')
plt.ylabel('Number of patterns')
plt.title('Number of patterns Vs Measure')
plt.scatter(hconf,length,label="H-confidence")
plt.scatter(support,length,label="support")
plt.legend()
plt.show()
```



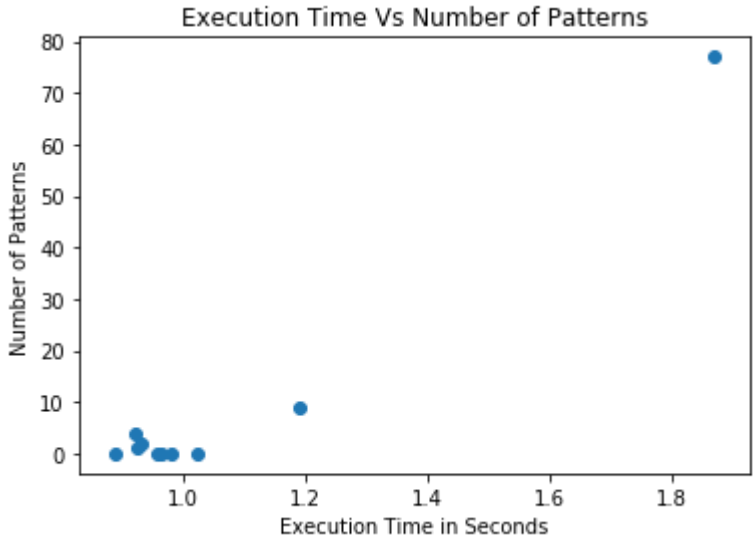
In [32]:

```
plt.xlabel('Measure')
plt.ylabel('Execution Time in Seconds')
plt.title('Execution Time Vs Measure')
plt.scatter(hconf,clock,label="H-confidence")
plt.scatter(support,clock,label="support")
plt.legend()
plt.show()
```



In [33]:

```
plt.xlabel('Execution Time in Seconds')
plt.ylabel('Number of Patterns')
plt.title('Execution Time Vs Number of Patterns')
plt.scatter(clock,length)
plt.show()
```



In [ ]: