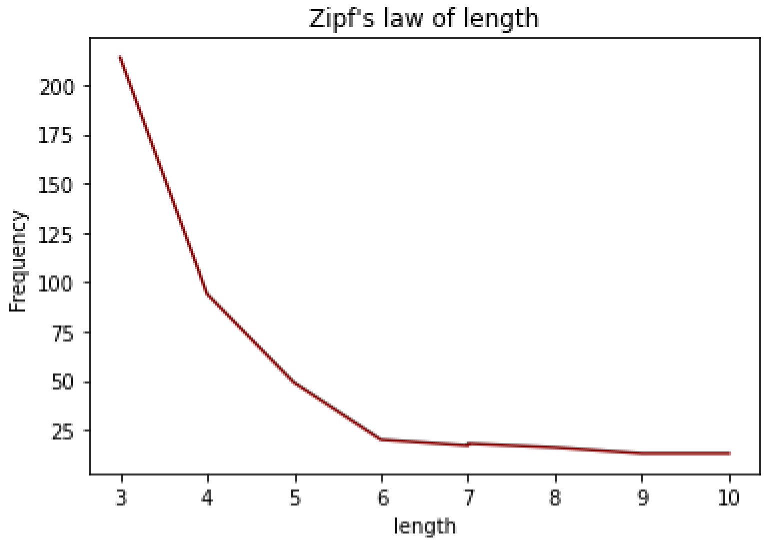
In [1]:
```
import csv
import nltk
import numpy as np
import re
import random
from nltk.corpus import wordnet
from nltk.tokenize import TweetTokenizer as tt
import math
tokens=[]
d=dict()
with open(r'tweets-dataset.csv','r',encoding="utf8") as csvfile:
    reader1=csv.reader(csvfile)
    c=0
    for i in reader1:
        token=tt().tokenize(i[0])
        actual=[]
        for i in token:
            k=re.findall(r"[A-Za-z']+",i)
            if len(k)==1 and len(i)==len(k[0]):
                if 'http' in k[0]:
                    k=[k[0].replace('http','')]
                if 'https' in k[0]:
                    k=[k[0].replace('https','')]
                k=[k[0].lower()]
                if not ((len(k[0])==1 and (k[0]=="'" )) or len(k[0])==0):
                    actual+=k
                if k[0] in d:
                    d[k[0]]+=1
                else:
                    d[k[0]]=1
        tokens+=actual
types=set(tokens)
n_tokens=len(tokens)
n_types=len(types)
print('Number of tokens : '+str(n_tokens))
print('Number of types : '+str(n_types))
print('TTR : '+str(n_types/n_tokens))
```

```
Number of tokens : 283330
Number of types : 30260
TTR : 0.10680125648537042
```

Zipf's Law of Length

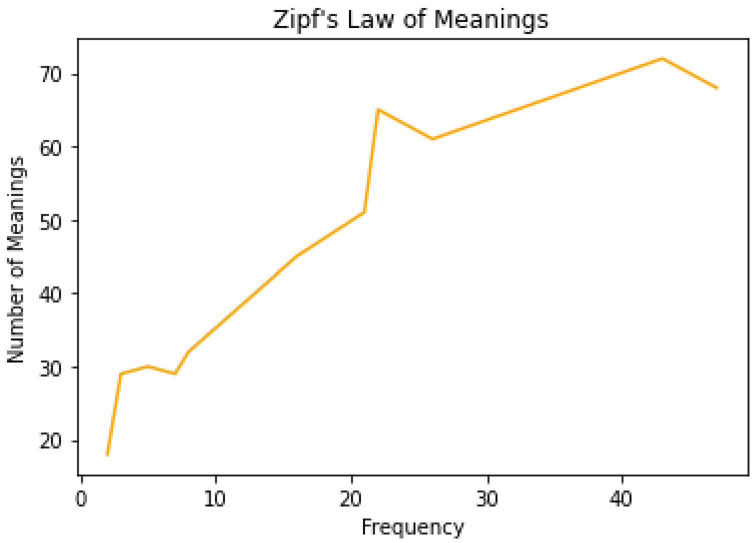
In [2]:
```
import matplotlib.pyplot as plt
%matplotlib inline
frequency_types=dict()
for i in tokens:
    if i in frequency_types:
        frequency_types[i]+=1
    else:
        frequency_types[i]=1
#wordlength_frequency=dict()
words_chosen=['show','complete','welcome','famous','holiday','champions','c

axes=[]
for i in words_chosen:
    axes.append([len(i),frequency_types[i],i])
axes.sort()
x=[]
y=[]
for i in axes:
    x.append(i[0])
    y.append(i[1])

plt.plot(x,y,color='maroon')
plt.title("Zipf's law of length")
plt.xlabel('length')
plt.ylabel('Frequency')
plt.show()
```



According to this law $frequency * wordlength = constant$, In the plot below the graph traces hyperbola $xy = c$, from which we can say that Zipf's law of length holds good.

Zip's Law of Meanings

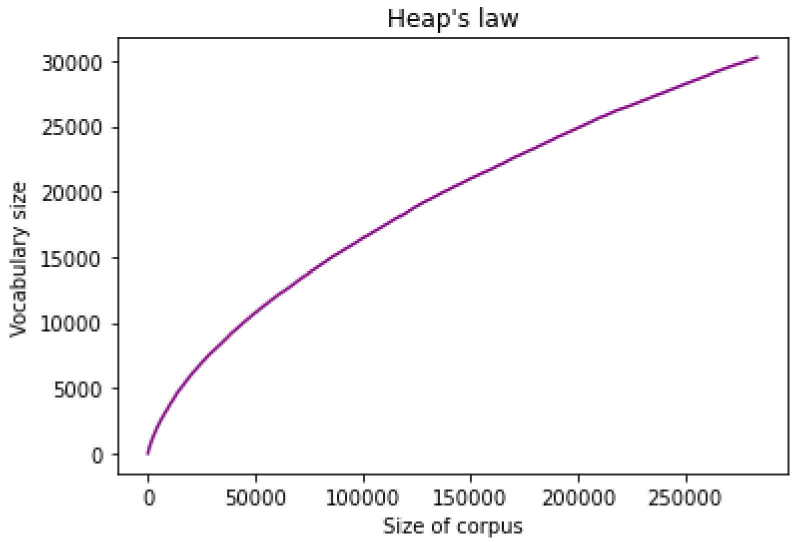
In [3]:
```
l1=[]
freq=[]
num_meanings=[]
words=['clubs','crackers','tip','matches','cases','starts','cover','deal','
meanings=[]
tokennumber=0
for i in words:
    c=0
    mean_i=[]
    for syn in wordnet.synsets(i):
        for j in syn.lemmas():
            mean_i.append(j.name())
    meanings.append(mean_i)
    num_meanings.append(len(set(mean_i)))
    freq.append(frequency_types[i])

plt.plot(freq,num_meanings,color='orange')
plt.title("Zipf's Law of Meanings")
plt.xlabel('Frequency')
plt.ylabel('Number of Meanings')
plt.show()
```



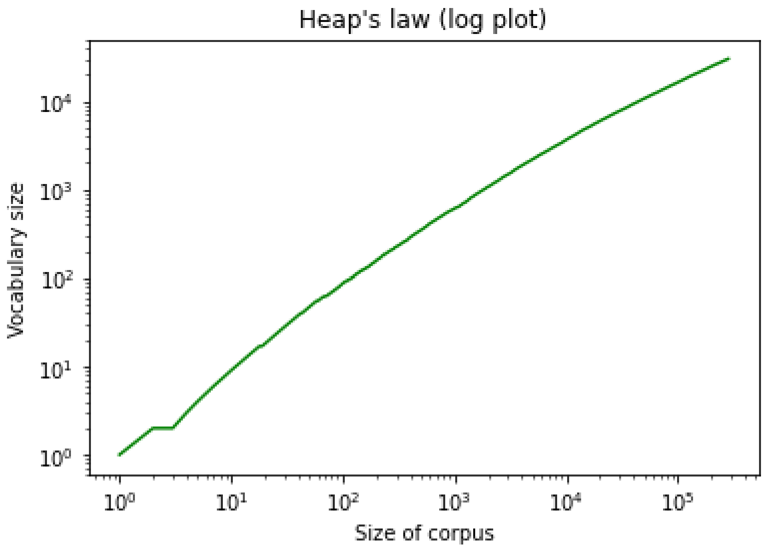
This plot seems to be tracing the curve $y = k * sqrt(x)$ so we can say that Zipf's law of meanings holds good

Heap's Law

In [4]:
```
hash_map={}
x=[]
y=[0]
for i in range(len(tokens)):
    x.append(i+1)
    if tokens[i] not in hash_map:
        y.append(y[-1]+1)
        hash_map[tokens[i]]=1
    else:
        y.append(y[-1])
plt.plot(x,y[1:],color='purple')
plt.title("Heap's law")
plt.xlabel('Size of corpus')
plt.ylabel('Vocabulary size')
plt.show()
base=[10 for i in range(len(x))]
a=np.log(x)/np.log(base)
b=np.log(y[1:])/np.log(base)
linfit=np.polyfit(a,b,1)
print('k = '+str(10**linfit[1])+' and '+'β = '+str(linfit[0]))
plt.plot(x,y[1:],color='green')
plt.xscale('log',basex=10)
plt.yscale('log',basey=10)
plt.title("Heap's law (log plot)")
plt.xlabel('Size of corpus')
plt.ylabel('Vocabulary size')
plt.show()
```




```
k = 9.327285751491175 and β = 0.6475121241458762

<ipython-input-4-ac30748407e6>:22: MatplotlibDeprecationWarning: The 'bas
ex' parameter of __init__() has been renamed 'base' since Matplotlib 3.3;
support for the old name will be dropped two minor releases later.
  plt.xscale('log',basex=10)
<ipython-input-4-ac30748407e6>:23: MatplotlibDeprecationWarning: The 'bas
ey' parameter of __init__() has been renamed 'base' since Matplotlib 3.3;
support for the old name will be dropped two minor releases later.
  plt.yscale('log',basey=10)
```



The plot of **log(vocabulary_size) vs log(size_of_corpus)** seems to be almost **linear** hence we can say that **Heap's Law** is satisfied. According to Heap's law $V = kN^{\beta}$ where V=size of vocabulary and N=size of the corpus. Here $k = 9.327$ and $\beta = 0.64751$

In [ ]: