In [1]:
```
import string
import matplotlib.pyplot as plt
from math import log
```

Read the datafile

In [2]:
```
f = open("romeoandjuliet.txt")
text = f.read()
f.close()
```

Create dictionary

In [3]:
```
d = {}
words = []
punc = string.punctuation #list of punctuations
content = text.lower().split() #make all words lowercase, split text into
```

Word processing

In [4]:
```
for w in content: #word processing
    if w[-1] in punc: # if ends with punc, remove
        new_w = w[:len(w)-1]
        words.append(new_w)
    elif w[0] in punc: # if begins with punc, remove
        new_w = w[1:]
        words.append(new_w)
    else: # otherwise, add
        words.append(w)
```

In [5]:
```
for w in words: #add and count words to dictionary
    if w in d:
        d[w] += 1
    else:
        d[w] = 1
```

Bonus: sort the dictionary according to decreasing word count.

In [6]:
```
sorted_d = sorted(d.items(), reverse=True, key=lambda x:x[1]) #sorted in
```

Plot a bar graph that shows the frequency with which each word appears.

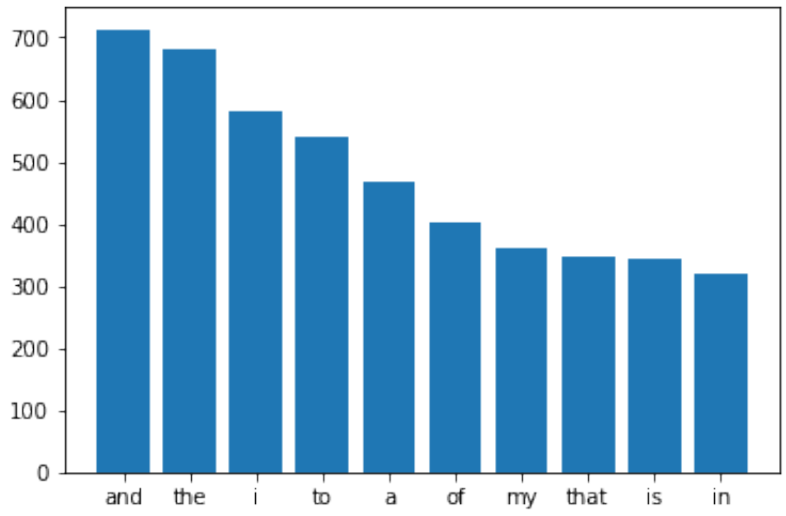
In [7]:
```
top_ten = sorted_d[:10] #the ten most frequent words
bottom_ten = sorted_d[-10:] #the least frequent

x = [x[0] for x in sorted_d]
y = [y[1] for y in sorted_d]
```

In [13]:
```
n_words = [i for i in range(len(x))]
plt.figure(num=None, figsize=(1000, 10), dpi=50)
plt.xlim(xmax=len(n_words))
plt.xlabel("words")
plt.ylabel("frequencies")
plt.bar(n_words,y)
plt.show()
#double click the graph below to view the complete graph
```

In this bar plot, the x axis indicates the ranks of words based on their frequencies, and are sorted in a descending pattern; the y axis indicates the number of times the word appears in the text. This graph shows that most of the words in the text appear only few times. The plot below has a higher resolution that demonstrates the top ten most-frequent words.

In [14]:
```
plt.bar(x[:10], y[:10])
plt.show()
```

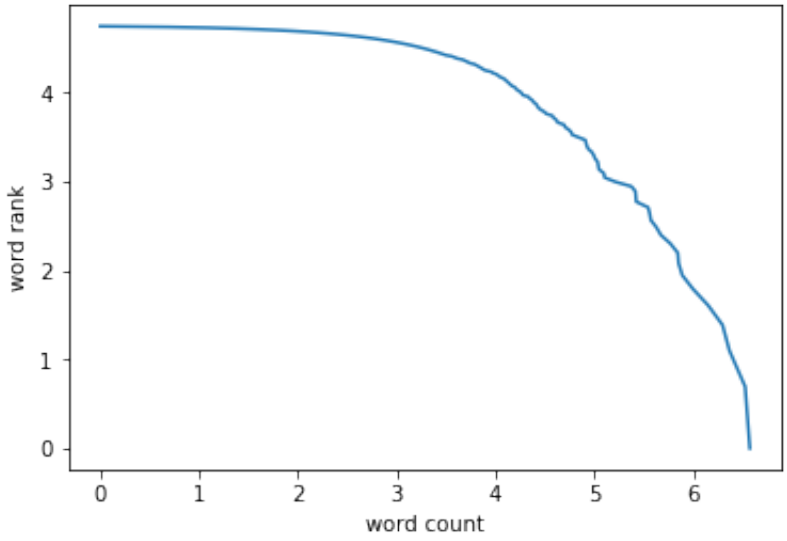


The words in x-axis are the ten most frequent words ("and" is the first most frequent, "in" is the tenth most frequent). The y-coordinates are number of counts of these words.

In [10]:
```
d_rank = {}
r = 1
for n in y: #loop through the sorted number of counts
    if n not in d_rank:
        d_rank[n] = r #assign ranks to the unique counts, starting from
        r += 1
```

In [11]:
```
# log[word count] and log[rank]
rank = [log(v) for v in d_rank.values()]
count = [log(k) for k in d_rank.keys()]
```

In [12]:
```
plt.plot(count, rank)
plt.xlabel('word count')
plt.ylabel('word rank')
plt.show()
```



What does this look like? Can you try to interpret this?

The log graph demonstrates that words with with more counts (appear more frequently) will have lower-number ranking (in this case, "lower-number ranking" means ranks closer to rank No.1).

Most of the words appear less frequently (many of the words only showed up 1 or 2 times in the text), therefore most of them have higher-number ranks as shown in the flat region.

As shown in the previous plot (the top-ten plot), the counts of the most frequent words descend so quickly. This explains the greater slope near the lower-ranking region.

In [ ]: