In [1]:

```
import numpy as np
import pandas as pd
```

In [7]:

```
#导入数据 271 343行的单词不在单词库里面，清洗掉
freqData=pd.read_csv("wordZip.csv")
freqData
df=pd.read_csv("cleaned_2_17.csv")
df.drop([271],inplace=True)
df.drop([343],inplace=True)
df.reset_index(drop=True, inplace=True)
df
```

Out[7]:

	nan	Date	Contest number	Word	Number of reported results	Number in hard mode	1 try	2 tries	3 tries	4 tries	5 tries	6 tries
0	NaN	2022/1/7 0:00	202	slump	80630	1362	1	3	23	39	24	9
1	NaN	2022/1/8 0:00	203	crank	101503	1763	1	5	23	31	24	14
2	NaN	2022/1/9 0:00	204	gorge	91477	1913	1	3	13	27	30	22
3	NaN	2022/1/10 0:00	205	query	107134	2242	1	4	16	30	30	17
4	NaN	2022/1/11 0:00	206	drink	153880	3017	1	9	35	34	16	5
...	...	...	...	...	...	...	...	...	...	...	...	...
352	NaN	2022/12/27 0:00	556	condo	20879	2012	0	2	17	35	29	14
353	NaN	2022/12/28 0:00	557	impel	20160	1937	0	3	21	40	25	9
354	NaN	2022/12/29 0:00	558	havoc	20001	1919	0	2	16	38	30	12
355	NaN	2022/12/30 0:00	559	molar	21204	1973	0	4	21	38	26	9
356	NaN	2022/12/31 0:00	560	manly	20380	1899	0	2	17	37	29	12

357 rows × 17 columns

In [16]:

```
#字母位次出现信息统计
#构建dataframe
wordData=freqData["word"]
letterSum=pd.DataFrame(columns=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p'
letterSum.to_csv("letterSum.csv")
```

In [40]:

```
letterSum=pd.read_csv("letterSum.csv")
letterSum
```

Out[40]:

	letter	one	two	three	four	five
0	a	0	0	0	0	0
1	b	0	0	0	0	0
2	c	0	0	0	0	0
3	d	0	0	0	0	0
4	e	0	0	0	0	0
5	f	0	0	0	0	0
6	g	0	0	0	0	0
7	h	0	0	0	0	0
8	i	0	0	0	0	0
9	j	0	0	0	0	0
10	k	0	0	0	0	0
11	l	0	0	0	0	0
12	m	0	0	0	0	0
13	n	0	0	0	0	0
14	o	0	0	0	0	0
15	p	0	0	0	0	0
16	q	0	0	0	0	0
17	r	0	0	0	0	0
18	s	0	0	0	0	0
19	t	0	0	0	0	0
20	u	0	0	0	0	0
21	v	0	0	0	0	0
22	w	0	0	0	0	0
23	x	0	0	0	0	0
24	y	0	0	0	0	0
25	z	0	0	0	0	0

In [41]:

```
#开始导入数据     ord('a')=97
for i in range(len(wordData)):
    for j in range(5):
            letterSum.iloc[ord(wordData[i][j])-97,j+1]=letterSum.iloc[ord(wordData[i][j])-97,j
letterSum
```

Out[41]:

	letter	one	two	three	four	five
0	a	140	304	306	162	63
1	b	173	16	56	24	11
2	c	198	40	56	150	31
3	d	111	20	75	69	118
4	e	72	241	177	318	422
5	f	135	8	25	35	26
6	g	115	11	67	76	41
7	h	69	144	9	28	137
8	i	34	201	266	158	11
9	j	20	2	3	2	0
10	k	20	10	12	55	113
11	l	87	200	112	162	155
12	m	107	38	61	68	42
13	n	37	87	137	182	130
14	o	41	279	243	132	58
15	p	141	61	57	50	56
16	q	23	5	1	0	0
17	r	105	267	163	150	212
18	s	365	16	80	171	36
19	t	149	77	111	139	253
20	u	33	185	165	82	1
21	v	43	15	49	45	0
22	w	82	44	26	25	17
23	x	0	14	12	3	8
24	y	6	22	29	3	364
25	z	3	2	11	20	4

In [44]:

```
#数据归一化，以列为单位归一
letterSum["one"]=letterSum["one"]/np.sum(letterSum['one'])
letterSum["two"]=letterSum["two"]/np.sum(letterSum['two'])
letterSum["three"]=letterSum["three"]/np.sum(letterSum['three'])
letterSum["four"]=letterSum["four"]/np.sum(letterSum['four'])
letterSum["five"]=letterSum["five"]/np.sum(letterSum['five'])
letterSum
```

Out[44]:

	letter	one	two	three	four	five
0	a	0.060632	0.131659	0.132525	0.070160	0.027285
1	b	0.074924	0.006929	0.024253	0.010394	0.004764
2	c	0.085751	0.017324	0.024253	0.064963	0.013426
3	d	0.048073	0.008662	0.032482	0.029883	0.051104
4	e	0.031182	0.104374	0.076657	0.137722	0.182763
5	f	0.058467	0.003465	0.010827	0.015158	0.011260
6	g	0.049805	0.004764	0.029017	0.032915	0.017757
7	h	0.029883	0.062365	0.003898	0.012126	0.059333
8	i	0.014725	0.087051	0.115201	0.068428	0.004764
9	j	0.008662	0.000866	0.001299	0.000866	0.000000
10	k	0.008662	0.004331	0.005197	0.023820	0.048939
11	l	0.037679	0.086618	0.048506	0.070160	0.067129
12	m	0.046340	0.016457	0.026418	0.029450	0.018190
13	n	0.016024	0.037679	0.059333	0.078822	0.056301
14	o	0.017757	0.120832	0.105240	0.057168	0.025119
15	p	0.061065	0.026418	0.024686	0.021654	0.024253
16	q	0.009961	0.002165	0.000433	0.000000	0.000000
17	r	0.045474	0.115634	0.070593	0.064963	0.091815
18	s	0.158077	0.006929	0.034647	0.074058	0.015591
19	t	0.064530	0.033348	0.048073	0.060199	0.109571
20	u	0.014292	0.080121	0.071460	0.035513	0.000433
21	v	0.018623	0.006496	0.021221	0.019489	0.000000
22	w	0.035513	0.019056	0.011260	0.010827	0.007362
23	x	0.000000	0.006063	0.005197	0.001299	0.003465
24	y	0.002599	0.009528	0.012560	0.001299	0.157644
25	z	0.001299	0.000866	0.004764	0.008662	0.001732

In [45]:

```
letterSum.to_csv("letterData.csv")
```

In [47]:

```
#给每个单词赋分:
score=[]
for i in range(len(wordData)):
    temp=[]
    for j in range(5):
        temp.append(letterSum.iloc[ord(wordData[i][j])-97,j+1])
    score.append(np.sum(temp))
freqData["score"]
freqData
```

Out[47]:

	word	freq	score
0	aback	9.348000e-07	0.313989
1	abase	9.217250e-08	0.456908
2	abate	1.533740e-06	0.443049
3	abbey	2.024110e-06	0.387181
4	abbot	2.140390e-06	0.258553
...	...	...	...
2304	young	1.668710e-04	0.291468
2305	youth	3.737070e-05	0.314422
2306	zebra	5.560490e-07	0.222174
2307	zesty	5.278750e-08	0.358164
2308	zonal	9.923600e-07	0.318753

2309 rows × 3 columns

In [51]:

```
# 导出与规范数据
letterScore=freqData.sort_values(by="score",ascending=False)
letterScore.reset_index(drop=True, inplace=True)
letterScore.to_csv('letterScore.csv')
```

In [179]:

```
#每个单词需要跑的数量 一共五十万个
a=freqData["score"]/np.sum(freqData["score"])
b=np.round(a*500000)
b
```

Out[179]:

```
0       174.0
1       253.0
2       245.0
3       214.0
4       143.0
        ...
2304    161.0
2305    174.0
2306    123.0
2307    198.0
2308    176.0
Name: score, Length: 2309, dtype: float64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [69]:

```
#蒙特卡洛分析
np.random.seed(0)
```

In [53]:

```
#概率赋值函数 把一个列表（得分）的值变成概率
def probability(df):
    score=np.array(df["score"])
    prob=np.divide(score,np.sum(score))
    for i in range(1):
        index = np.random.choice(np.array(df["word"]), p=prob.ravel())
    return index
```

In [62]:

```
#检测两个单词是否相等
def checkEnd(myWord, problemWord):
    return myWord==problemWord
```

In [141]:

```
#输入旧单词 弹出新单词 选择列表中的单词取决于单词的概率
def newWord(problemWord):
    wordRange=letterScore.copy(deep=True)
    wordRange["index"]=range(len(wordRange))
    deadCount=0
    conditionOUT=[]
    chosen=probability(wordRange)

    while(checkEnd(chosen,problemWord)!=True):
        for i in range(5):
            if (chosen[i] == problemWord[i]):
                wordRange=wordRange[wordRange["word"].str[i]==chosen[i]]
            elif(chosen[i] in problemWord):
                if(chosen[i] in conditionOUT):
                    pass
                else:
                    wordRange=wordRange[wordRange["word"].str.contains(chosen[i])]
                    conditionOUT.append(chosen[i])

        conditionOUT=[]
        deadCount=deadCount+1
        if(deadCount==6):
            return [7,chosen,False]

        chosen=probability(wordRange) #重新选择chosen单词
        wordRange=wordRange[wordRange["word"]!=chosen]
        wordRange.reset_index(drop=True, inplace=True)
    return [deadCount+1,chosen,True]
#     print(count)
#     print(chosen)
```

In [142]:

```
#随机生成问题:
problemWord=wordData[np.random.randint(0,num)]
```

In [143]:

```
# 创建单词集合
dfMont=pd.DataFrame({"word":wordData,'one':0,'two':0,'three':0,'four':0,'five':0,'six':0,'seven
dfMont
```

Out[143]:

	word	one	two	three	four	five	six	seven	False
0	aback	0	0	0	0	0	0	0	0
1	abase	0	0	0	0	0	0	0	0
2	abate	0	0	0	0	0	0	0	0
3	abbey	0	0	0	0	0	0	0	0
4	abbot	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
2304	young	0	0	0	0	0	0	0	0
2305	youth	0	0	0	0	0	0	0	0
2306	zebra	0	0	0	0	0	0	0	0
2307	zesty	0	0	0	0	0	0	0	0
2308	zonal	0	0	0	0	0	0	0	0

2309 rows × 9 columns

In [ ]:

In [178]:

```
# #随机生成问题:
num=len(dfMont)
for i in range(len(b)):
#     problemWord=dfMont["word"][np.random.randint(0,num)] #这是随机生成
    for j in range(int(b[i])):
        problemWord=dfMont["word"][i]
        chosenWord=newWord(problemWord)
        dfMont.iloc[dfMont[dfMont["word"]==chosenWord[1]].index[0],chosenWord[0]]=dfMont.iloc[d
        if(chosenWord[2]==False):
            dfMont.iloc[dfMont[dfMont["word"]==problemWord].index[0],8]=dfMont.iloc[dfMont[dfMo

dfMont
```

Out[178]:

	word	one	two	three	four	five	six	seven	False
0	aback	0	0	1	1	0	0	0	1
1	abase	0	0	0	0	1	0	3	1
2	abate	0	0	0	1	1	0	3	1
3	abbey	0	0	0	0	0	0	0	1
4	abbot	0	0	0	0	0	0	1	1
...	...	...	...	...	...	...	...	...	...
2304	young	0	0	0	1	0	0	0	2
2305	youth	0	0	0	0	0	0	2	3
2306	zebra	0	0	0	0	0	0	1	1
2307	zesty	0	0	0	0	1	0	0	2
2308	zonal	0	0	0	0	0	1	1	1

2309 rows × 9 columns

In [134]:

```
dfMont[dfMont["word"]=='impel'].index[0]
```

Out[134]:

```
353
```

In [168]:

```
int(b[i])
```

Out[168]:

```
1
```

In [ ]:

```

```