断句

断句是困难的，因为句号会被用来标记缩写而成另一些句号同时标记缩写，如U.S.A。

分词

对于一些书写系统，由于没有词边界的可视表示这一事实，文本分词变得更加困难。

在中文中，3个字符的字符串：爱国人，是“爱，国人”还是“爱国，人”，确实困难。

给一段 去掉边界的英文来分词

**text = "doyouseethekittyseethedoggydoyoulikethekittylikethedoggy"**

**最开始，问题是在什么时候切，或者说在哪个字母后面切，这样，用1表示在这个字母后面断开，就把文本分词可以用二进制表示了。  
>>> seg1 = "0000000000000001000000000010000000000000000100000000000"  
>>> seg2 = "0100100100100001001001000010100100010010000100010010000"**

seg1和seg2是两种不同的序列，代表不同的分词。

现在的问题就是哪些地方应该是1。

原书给出了一个简单的方式判别一个分词方式好不好，就是1.单词数加2.生成的句子的字符和空格数的总和最小。

然后作者用随机的方式，即模拟退火算法来找总和最小的序列。

对于这样一个算法，我表示很怀疑，这个简单甚至可以说粗暴的方式我明白其中的科学性有多大，在我看来只是一种经验方法。完全没有得到正确结果的必然。

而且作者初始化序列为seg1，这是明显的人工干预。如果初始化为全0序列，结果不令人满意。

1.(43, ['doyou', 'se', 'ethekitty', 'se', 'ethedoggy', 'doyou', 'lik', 'ethekitty', 'lik', 'ethedoggy'])

2.(43, ['doyou', 'seet', 'hekitty', 'seet', 'hedoggy', 'doyou', 'liket', 'hekitty', 'liket', 'hedoggy'])

实验了几次，虽然很接近可接受的结果，但是结果使用模拟退火算法结果很不稳定，在我看来，整个算法有很大提升空间。

不过，作者算是提供了一个解决此类问题的一个流程，只是算法待设计。

**def segment(text, segs):  
 words = []  
 last = 0  
 for i in range(len(segs)):  
 if segs[i] == '1':  
 words.append(text[last:i+1])  
 last = i+1  
 words.append(text[last:])  
 return words**

**def evaluate(text, segs):  
 words = segment(text, segs)  
 text\_size = len(words)  
 lexicon\_size = sum(len(word) + 1 for word in set(words))  
 return text\_size + lexicon\_size**

**from random import randint  
  
def flip(segs, pos):  
 return segs[:pos] + str(1-int(segs[pos])) + segs[pos+1:]  
  
def flip\_n(segs, n):  
 for i in range(n):  
 segs = flip(segs, randint(0, len(segs)-1))  
 return segs**

def anneal(text, segs, iterations, cooling\_rate):

temperature = float(len(segs))

while temperature > 0.5:

best\_segs, best = segs, evaluate(text, segs)

for i in range(iterations):

guess = flip\_n(segs, int(round(temperature)))

score = evaluate(text, guess)

if score < best:

best, best\_segs = score, guess

score, segs = best, best\_segs

temperature = temperature / cooling\_rate

print(evaluate(text, segs), segment(text, segs))

print()

return segs