Алгоритм построения всех реверсивных текстов в 3 слова по заданному словарю

1. По словарю строим хэш (ассоциативный массив), где ключом являются всевозможные начала слов (длиной до 8 букв), а значение будет массив слов, где ключ является началом.
Назовём его wbeg.

2. По словарю строим хэш (ассоциативный массив), где ключом являются всевозможные инвертированные концы слов (длиной до 8 букв), а значение будет массив сло.в, где встречается такой конец слов.
Назовём его wend.

3. Рассмотри возможные варианты структуры реверсивного текста из 3 слов.
Обозначим три слова как W1 W2 W3 и их длины L1, L2, L3.

Вариант A. L3 > L1.
В этом случае W3 можно представить как W3 = W31 + W32,
где W32 = Rev(W1), где Rev() функция реверсирования текста.
А W2+W31 - должен быть палиндромом. Пусть L31 длина W31.
Вариант A1. L2 > L31 (L1+L2>L3).
В этом случае W2 = W21 + W22, где W21 = Rev(W31), а W22 - палиндром,
то есть W22 = Rev(W22).
Вариант A2. L2 < L31
В этом случае W31 = W311 + W312, где W2 = Rev(W312), а W311 - палиндром,
то есть W311 = Rev(W311)
Вариант A3. L2 == L31
В этом случае W2 = Rev(W31)

Вариант B. L1 > L3.
В этом случае W1 можно представить как W1 = W11 + W12,
где W11 = Rev(W3).
А W12+W2 - должен быть палиндромом. Пусть L12 длина W12.
Вариант B1. L12 > L2
В этом случае W12 = W121 + W122, где W121 = Rev(W2), а W122 - палиндром,
то есть W122 = Rev(W122)
Вариант B2. L12 < L2
В этом случае W2 = W21 + W22, где W12 = Rev(W22), а W21 - палиндром,
то есть W21 = Rev(W21). Или также палиндромом будет W12+W2.
Можно проверять или то или другое на выбор.
Вариант B3. L12 == L2
В этом случае W12 = Rev(W2)

Вариант C. L1 == L3.
В этом случае W1 = Rev(W3) и W2 палиндром, то есть это составной палиндром, состоящий как 'матрёшка' из двух палинлромов W1+W3 и W2.

4. Алгоритм на языке Python

```
import os
import re
import time

# файл нежелательных слов для фильтрации
wordfilters = set()
with open(os.getcwd()+"\\valfilters.txt",'rt') as fin :
```

```
  for line in fin :
    flt = line.strip()
    if (flt!="") : wordfilters.add(flt)

# файл авторских палиндромов
pal3 = set()
with open(os.getcwd()+"\\..\\data\\palbase3.txt",'rt') as fin :
  for line in fin :
    pal3.add(line.rstrip())

print("Читаем словарь wordict.txt...")
dict = {} # словарь
wbeg = {} # хэш по началау слов
wend = {} # хэш по инвертированному концу слов
LSMAX = 8 # максимальнная длина начала и конца слов (wbeg,wend)
        # равна максмимальной длине слова, которое будучи реверсированным
        # может быть началом или концом другого слова. Из опыта - это 8

with open(os.getcwd()+"\\..\\data\\wordict_short.txt",'rt',encoding='UTF-8') as f :
  for line in f :
    ls = line.strip().split(";")
    w = ls[0].replace('-',").replace('ё','e')
    if (w in wordfilters) : continue
    if (w in dict) : continue
    dict[w] = 1
    lw = len(w)
    for i in range(1,min(lw+1,LSMAX)) :
      wb = w[:i];
      if (wb not in wbeg) : wbeg[wb] = {}
      wbeg[wb][w] = 1
      we = w[:-(i+1):-1];
      if (we not in wend) : wend[we] = {}
      wend[we][w] = 1

palind = set()
print("Генерируем палиндромы...")
for wc in sorted(dict) :
  # print(wc)
  lw = len(wc)
  if (lw>LSMAX) : continue
  # Вариант A: L3>L1 первое слово самое короткое, третье - самое длинное.
  w1 = wc
  if (w1 in wend) :
    for w3 in wend[w1] :
      rw31 = w3[::-1][lw:]
      # Вариант A.1: L1+L2>L3
      if (rw31 in wbeg) :
        l31 = len(rw31)
        for w2 in wbeg[rw31] :
          w22 = w2[l31:]
          if (w22==w22[::-1]) :
            p = w1+' '+w2+' '+w3
            pr = '*' if (p in pal3) else "
            palind.add(pr+p)
      # Вариант A.3 : L1+L2 == L3
      if (rw31 in dict) :
        p = w1+' '+rw31+' '+w3
```

```
          pr = '*' if (p in pal3) else ''
          palind.add(pr+p)
        # Вариант A.2 : L1+L2<L3
        # строим все варианты расщепления rw31
        # где начальный отрезок и будет w2
        # а конечная часть должны быть палиндромом
        for i in range(1,len(rw31)) :
          w2 = rw31[:i]
          if (w2 in dict) :
            rw311 = rw31[i:]
            if (rw311==rw311[::-1]) :
              p = w1+' '+w2+' '+w3
              pr = '*' if (p in pal3) else ''
              palind.add(pr+p)
    w3 = wc
    # вариант B: L1>L3, первое слово длиннее третьего
    rw3 = w3[::-1] # реверс третьего слова - начало первого слова w1
    if (rw3 in wbeg) :
      for w1 in wbeg[rw3] :
        w12 = w1[lw:]
        rw = w12[::-1]
        # если реверс остатка первого слова (rw) является словом словаря,
        # то имеем палиндром (w1,w2,w3), где w2 = rw
        # Подвариант B.3: L2+L3=L1
        if (rw in dict) :
          p = w1+' '+rw+' '+w3
          pr = '*' if (p in pal3) else ''
          palind.add(pr+p)
        # делим остаток w12 на две части
        # Подвариант B.1 L2+L3<L1
        for i in range(0,len(w12)) :
          w2 = w12[:i][::-1] # реверс первой части остатка
          # если реверс первой части остатка первого слова является словом словаря
          if (w2 in dict) :
            w122 = w12[i:]
            # а вторая часть остатка - палиндром
            if (w122==w122[::-1]) :
              # то имеем палиндром (w1,w2,w3)
              p = w1+' '+w2+' '+w3
              pr = '*' if (p in pal3) else ''
              palind.add(pr+p)
        # Вариант B.2 L2+L3>L1
        if (w12 in wend) :
          for w2 in wend[w12] :
            ws = w12+w2
            if (ws==ws[::-1]) :
              p = w1+' '+w2+' '+w3
              pr = '*' if (p in pal3) else ''
              palind.add(pr+p)

with open(os.getcwd()+"\\..\\data\\palind3.txt",'wt',encoding='windows-1251') as fout :
  for p in sorted(palind) :
    fout.write(p+'\n')
```