principia palindromica

R.J. Toscani

ABSTRACT

Dit document heeft als onderwerp ***"principia palindromica"*** *en is het resultaat van een automatische conversie vanuit platte tekst naar het TROFF-formaat, door middel van het script* ***text2troff29.sh****, een project van Rob Toscani.*

18 March 2024

principia palindromica

R.J. Toscani

Beschouwing middenwoorden in palindroomzin

We definiëren de "Scheefheid" van een willekeurig woord als het aantal karakters dat uit de ***genormaliseerde representatie van het woord*** *moet worden verwijderd (hierna "scheve karakters") voordat bij het restant van de string een symmetrie wordt gevonden d.m.v. de vergelijking:*

```
string == string[::-1]
```

Dit symmetrische deel van het middenwoord wordt dan in het middelpunt van de te vinden palindroom-zin geplaatst.

Definities voor het kwantificeren van de "scheefheid":

- *Verwijderen vanaf begin van het woord (links) tot symmetrie levert een scheefheid met een negatief getal voor het aantal karakters op.*
- *Verwijderen vanaf eind van het woord (rechts) tot symmetrie levert een scheefheid met een positief getal voor het aantal karakters op.*

(Bovenstaande definities zijn arbitrair en zouden eventueel nog omgedraaid kunnen worden.)

Zo bezien kunnen bij elk woord minimaal 2 symmetrische strings worden gevonden, als er maar genoeg karakters als scheefheid worden verwijderd, nl.:

- *de enkele beginletter is symmetrisch, na verwijderen alle scheefheid vanaf rechts;*
- *de enkele eindletter is symmetrisch, na verwijderen scheefheid vanaf links.*

Zo'n begin- of eindkarakter kan dan in het midden van een palindroomzin geplaatst worden (respectievelijk de huidige typen 2 en 3). Het scheefheidsdeel steekt dan resp. naar rechts dan wel naar links uit.

Daarnaast hebben diverse woorden nog een of meer extra symmetrieeën na verwijderen van de overige scheefheid.

Als er bijvoorbeeld helemaal niets hoeft te worden verwijderd tot vinden van de eerste symmetrische string (ofwel de scheefheid is dan 0), dan is er sprake van zuivere symmetrie:

- *met één uniek middenkarakter:*

  ```
  "abcdefgfedcba"
  ```

- *met twee identieke middenkarakters:*

  ```
  "abcdefggfedcba"
  ```

En bij een woord als:

```
"abababababa"
```

zijn er daarnaast zowel vanaf links als vanaf rechts zelfs diverse overige symmetrieën te vinden.

Zoals hierboven al bleek is er 1-punts-symmetrie en 2-punts-symmetrie, ofwel 1 uniek middelpunt (oneven palindroomlengte) en twee identieke middelpunten naast elkaar (even palindroomlengte). (Strings met meer dan twee identieke middenkarakters zijn bijzondere gevallen van bovengenoemde twee.)

Bij een string als:

```
"aabcdeff"
```

hebben we b.v. twee symmetrieeën (een 1-punt en een 2-punts) aan zowel het begin als het eind, na verwijdering van de nodige scheefheid.

Kortom, elk woord herbergt dus in feite twee of meer palindroomwoorden, als er maar genoeg "scheve" karakters worden verwijderd.

Per woord kunnen alle scheefheden worden vastgelegd in een dictionary met als key het genormaliseerde woord en als value een lijst met gevonden scheefheidswaarden tot symmetrie.

En aldus lenen alle woorden zich om als middenwoord geplaatst te worden. De eventuele "scheve" karakters steken dan of meer naar links of meer naar rechts uit vanaf het middelpunt.

Als ze naar rechts meer uitsteken dan links, moeten de overige passende woordcombinaties eerst aan de "krappe" zijde dus rechts worden geplaatst – we noemen in dit geval rechts de "primaire zijde" – en dan naar links worden gespiegeld ("secundaire zijde", met aanvulling van de scheve karakters van het middenwoord daarna), waarna het resultaat daar wordt gepartioneerd.

Als ze naar links meer uitsteken dan rechts, moeten de overige passende woordcombinaties eerst aan de "krappe" zijde dus links worden geplaatst – nu is links de "primaire zijde"- en dan naar rechts worden gespiegeld ("secundaire zijde, met aanvulling van de scheve karakters van het middenwoord daarvoor), waarna het resultaat daar wordt gepartioneerd.

Na verificatie van de partities tegen de woordenlijst is de resulterende palindroomzin dan een samenvoeging van de combinatie van woorden aan de linker zijde, het middenwoord en de combinatie van woorden aan de rechter zijde.

Bij zuiver symmetrische middenwoorden kan naar beide richtingen worden gespiegeld voorafgaand aan partionering. Praktisch is dan alleen 1x dit proces uit te voeren, b.v. van links (primair) naar rechts (secudair) en daar partitioneren, en daarna de resultaten op 2 manieren presenteren: wel en niet verwisseld.

Bij geen middenwoord (huidige type 1), dus indien:

```
middenwoord-string == ""
```

geldt hetzelfde als bij zuiver symmetrische middenwoorden.

Of een (zuiver of scheef) middenwoord met lengte W en scheefheid S wel of niet past in de ingestelde palindroom(zin-)lengte L, dient als volgt vastgesteld te worden:

1. *Bij **1-punts symmetrie** (geeft palindroom-zin met oneven lengte):*

   ```
   (W - |S|) // 2 + |S| <=
   (L - 1) // 2
   ```

   *Deze formule wordt gebruikt als middenwoordlengte en scheefheid **tegengestelde parity** hebben (even vs. oneven) Let op: woordlengte W bepalen uit de genormaliseerde representatie van het woord.*

2. *Bij **2-punts symmetrie** (geeft palindroom-zin met even lengte):*

   ```
   (W - |S|) // 2 + |S| <=
   L // 2
   ```

   *Deze formule wordt gebruikt als middenwoordlengte en scheefheid **gelijke parity** hebben (beide even of oneven). Let op: woordlengte W bepalen uit de genormaliseerde representatie van het woord.*

De beschikbare lengte P van de woordcombinatie aan primaire zijde wordt dan:

```
P = ( L - W - |S| ) // 2
```

Het teken van S bepaalt of die eerste combinatie aan de linkerkant (S negatief) of de rechterkant (S positief) is. Let op: woordlengte W bepalen uit de genormaliseerde representatie van het woord.

De behandeling van zoekwoorden wordt nu als volgt: In geval van wel een middenwoord wordt de plaatsing van de zoekwoorden (de reeks non-option arguments) roulerend gemaakt:

- *Eerst het 1e woord op middenpositie (in al zijn symmetriestanden/scheefheden), de overige in volgorde voor zover passend op de primaire zijde (dat kan dus links of rechts zijn afhankelijk van de stand van het middenwoord);*
- *Daarna het 2e woord op de middenpositie, en de overige weer in volgorde op de primaire zijde;*
- *Vervolgens het 3e, met verder weer alles als hiervoor;*
- *Volgende woord, etc. etc. tot alle woorden zijn geweest.*

In geval van geen middenwoord komen alle zoekwoorden in volgorde voor zover passend altijd aan de linkerkant. Dat dan telkens weer gevolgd door permutatie en spiegeling naar rechts zoals nu ook al het geval is, waarna de resultaten onderling links ←→ rechts worden verwisseld (zie boven).

Hierbij wordt erkend dat de zoekwoorden sowieso geen compleet beeld zullen blijven opleveren maar slechts een benadering zijn en soms in verkeerde volgorde. Toch zullen door de roulering in combinatie met de permutaties naar verwachting ook voldoende combinaties worden aangeboden in een juiste volgorde.

Wat betekent dit alles nu voor het algoritme van palindromes.py ?

1. *Van alle woorden alle mogelijke scheefheden verzamelen d.m.v. een nieuwe functie of uitgebreide versie van de functie "test_palindrome()". De scheefheden bepalen uit de genormaliseerde representatie van de woorden.*

   → *Invoegen in: het blok op regels 383 – 398*

2. *Hiervoor een nieuwe dictionary bijhouden, met als key het woord (niet genormaliseerd!) en als value een lijst met alle scheefheids-waarden, uitgedrukt in een negatief (links meer) of positief (rechts meer) heel getal, of 0 (zuiver symmetrisch).*

   → *Invoegen in: het blok op regels 383 – 398*

3. *Neem ook de woorden in de non_option_arguments mee in deze scheefheids-waarden-dictionary. Dit zijn namelijk mogelijk niet-bestaande woorden.*

   → *Invoegen in: het blok op regels 383 – 398*

4. *Het programma kiest afzonderlijk voor **geen** en **wel** een middenwoord, er zijn daarmee nu slechts nog 2 typen. In het laatste geval via een random gekozen index waarmee in (de reeds bestaande) dictlist_reduced wordt geselecteerd. Bij optie -S in alfabetische volgorde. Geen middenwoord betekent: middenwoordlengte en scheefheid = 0*

   → *Invoegen in: Dit wordt een toegevoegde loop over het blok v/a regel 400 tot eind heen. De indeling van functie generate_results() hierop ook aanpassen (o.m. de type-indeling).*

5. *Indien er woord-argumenten ("non_option_args") zijn gegeven, gaan deze in de **geen middenwoord** modus alle in ongewijzigde volgorde naar het begin van de primaire zijde (voor zover passend). Dat is bij geen middenwoord altijd de linker zijde.*

   → *Invoegen in: binnen de loop van (4.)*

6. *In de **wel middenwoord** modus daarentegen gaat een nieuwe functie d.m.v. een loop door de reeks woord-argumenten heen. Daarbij wordt telkens een ander woord van de reeks als middenwoord aangewezen en gaan de overige woorden in ongewijzigde volgorde naar het begin van de primaire zijde (voor zover passend).*

   → *Invoegen in: Dit wordt een toegevoegde loop binnen de loop van (4.)*

7. *Een loop gaat per middenwoord nu langs alle scheefheden ("standen"), en per geval bepaalt het teken welke zijde naast het middenwoord de primaire zijde (de "krappe zijde") is:*

   – *linkerzijde bij negatieve S*

   – *rechterzijde bij positieve S*

   → *Invoegen in: binnen de loop van (4.)*

8. *Tevens per stand bepalen of het middenwoord past binnen de gestelde palindroomzinlengte, zie eerste twee vergelijkingen hierboven, afhankelijk van de onderlinge parities van middenwoordlengte en scheefheid. Gebuik de genormaliseerde representatie van het woord om de lengte te bepalen. Past het niet, dan meteen naar volgende scheefheid/stand.*

   → *Invoegen in: binnen de loop van (7.)*

9. *De max_word_qty wordt in geval van passende middenwoord-stand gelijk aan geldende waarde minus 1.*

   → *Invoegen in: vervanger van regel 404, gekoppeld aan (8.) binnen de loop van (7.)*

10. *Per stand van het middenwoord de string_length bepalen die beschikbaar is voor de geldende primaire zijde (links of rechts), d.m.v. de derde vergelijking hierboven.*

    → *Invoegen in: vervanger van regel 401, gekoppeld aan (8.) binnen de loop van (7.)*

11. *Indien de lengte van (de genormaliseerde representatie van) een zuiver symmetrisch middenwoord gelijk is aan palindoomlengte cq qty = 1 blijven linker- en rechterdeel lege strings. Dan wordt alleen het middenwoord geprint als palindroomwoord, mits met optie -q = 1.*

    → *Is een bijzonder resultaat van de hier beschreven generieke functionaliteit.*

12. *Van middenwoord en primaire zijde worden genormaliseerde representaties opgehaald die worden gebruikt om via spiegeling de secundaire zijde te genereren.*

    → *Invoegen in: blok 406 e.v. of onderdeel van gemodificeerde functie generate_results()*

13. *Bij negatieve scheefheid worden evenzoveel genormaliseerde karakters van het **begin** van het middenwoord meegespiegeld met het **einde** van het linkerdeel. Resultaat is een rechterdeel met aangevuld **begin**, dat vervolgens wordt gepartitioneerd.*

    → *Invoegen in: blok 406 e.v. of onderdeel van gemodificeerde functie generate_results()*

14. *Bij positieve scheefheid worden evenzoveel genormaliseerde karakters van het **eind** van het middenwoord meegespiegeld met het **begin** van het rechterdeel. Resultaat is een linkerdeel met aangevuld **eind**, dat vervolgens wordt gepartitioneerd.*

    → *Invoegen in: blok 406 e.v. of onderdeel van gemodificeerde functie generate_results()*

15. *Bij geen of zuiver middenwoord (scheefheid = 0) alleen van primair links de genormaliseerde representatie ophalen en spiegelen naar secundair rechts, niet andersom.*

    → *Invoegen in: is al huidige functionaliteit, nu gestuurd door (4.) en (12.)*

16. *Haal van de genormaliseerde woordcombinaties in het secondaire deel alle passende woorden op (zoals dat reeds in het bestaande programma gebeurt).*

    → *Is al onderdeel van functie get_words(), mogelijk iets aanvullen.*

17. *Het resultaat wordt nu samengesteld uit linkerdeel + middendeel + rechterdeel. Bij geen of zuiver middenwoord*

worden daarna tevens nog het linker- en rechterdeel verwisseld als tweede resultaat.

→ *Invoegen in: functie get_ words()*