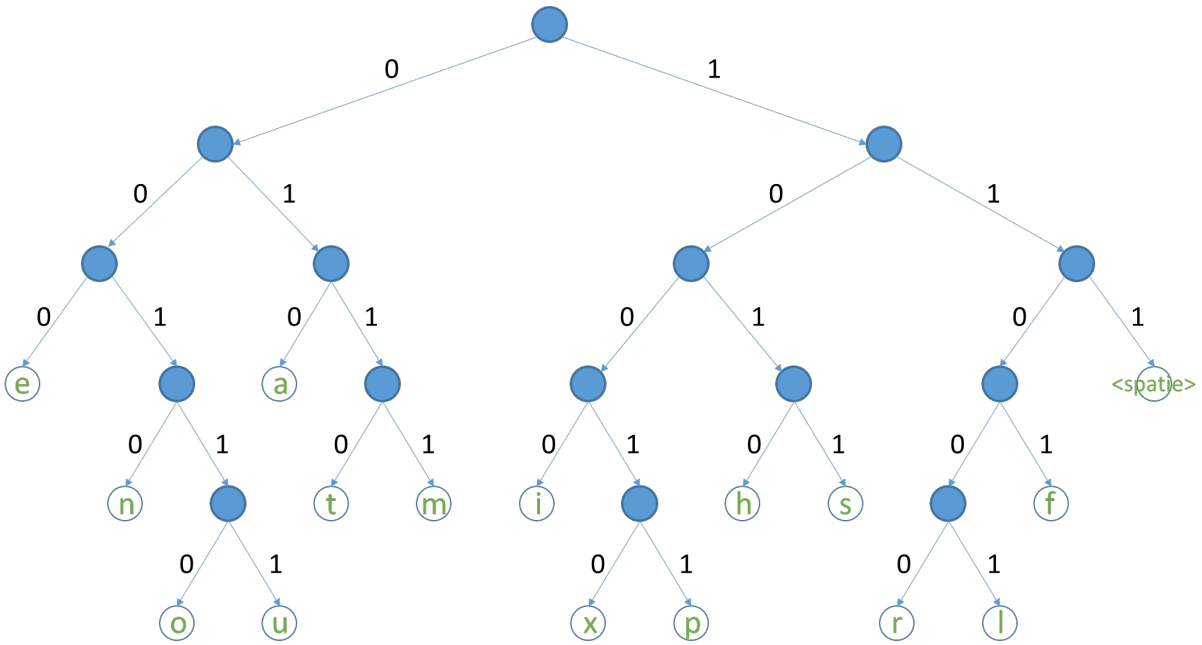
Huffman coding

Huffmancodering is een techniek uit de informatica die gebruikt wordt voor verliesloze datacompressie. Omdat gegevens binnen een computer moeten voorgesteld worden als **bitstrings** (strings van opeenvolgende nullen en enen), wordt elk symbool — bijvoorbeeld een karakter van een tekst — traditioneel voorgesteld door een bitstring met een vast aantal bits. Als we op die manier bijvoorbeeld elk karakter van het woord huffman voorstellen door één **byte** (8 bits), dan gebruikt de computervoorstelling van dit woord in totaal 56 bits.

Tijdens zijn doctoraat aan MIT ontwikkelde David A. Huffman een algoritme dat elk symbool voorstelt door een bitstring, waarbij het aantal bits verschilt per symbool. De details van dit algoritme zijn voor deze opgave niet belangrijk, maar de winst bij compressie zit hem in het feit dat symbolen die vaker voorkomen worden voorgesteld door kortere bitstrings dan symbolen die minder vaak voorkomen. Het resultaat van het algoritme is een **binaire boom** waarvan de linkertakken gelabeld zijn met bit 0 en de rechtertakken met bit 1. De interne knopen van de boom zijn niet gelabeld met symbolen, terwijl alle bladeren wel gelabeld zijn met een uniek symbool.



Huffmanboom gegenereerd aan de hand van het aantal voorkomens van de karakters in de tekst "*this is an example of a huffman tree*". De bitstringcodering van de zin aan de hand van deze code vereist 135 bits, daar waar er 288 bits nodig zijn als elk van de 36 karakters wordt voorgesteld door 8 bits.

Bij het **coderen** van een tekst als een bitstring wordt elk symbool afzonderlijk voorgesteld door een unieke bitstring. De bitstring van een symbool vind je door in de boom af te dalen vanaf de wortel naar het blad dat gelabeld is met het symbool, en de labels van de takken die je daarbij afloopt achter elkaar te zetten. Op basis van bovenstaande boom zien we bijvoorbeeld dat de letter h wordt voorgesteld door de bitstring 1010, de letter u door de bitstring 00111, de letter f door de bitstring 1101 en een spatie door de bitstring 111. De tekst huffman wordt volgens de codering uit bovenstaande boom dus voorgesteld door de bitstring 1010001111101110101110100010 (28 bits). Merk dus op dat we voor het coderen van een tekst de boom eigenlijk niet nodig hebben, aangezien elk symbool wordt voorgesteld door een vaste bitstring. Voor bovenstaande boom worden de

symbolen dus als volgt vertaald naar bitstrings.

symbool	bitstring	symbool	bitstring
spatie	111	s	1011
a	010	t	0110
e	000	l	11001
f	1101	o	00110
h	1010	p	10011
i	1000	r	11000
m	0111	u	00111
n	0010	x	10010

Het **decoderen** van een bitstring verloopt wel aan de hand van de boom. Stel bijvoorbeeld dat we de bitstring `10100011111011101010111010100010` willen decoderen, dan starten we bij de wortel van de boom, en dalen we telkens af langs de tak die aangegeven wordt door het volgende bit. Voor de voorbeeld bitstring dalen we dus eerst af naar rechts (bit `1`), dan naar links (bit `0`), dan naar rechts (bit `1`) en terug naar links (bit `0`) om uiteindelijk het blad van de boom te bereiken dat gelabeld is met de letter `h`. Van zodra we een blad bereikt hebben, gebruiken we het symbool waarmee het blad gelabeld is als volgende letter van de tekst. Daarna springen we terug naar de wortel van de boom, om vanaf daar verder te gaan met het verwerken van de bits. De bitstring `00111` levert dan de letter `u` op, tweemaal de bitstring `1101` levert tweemaal de letter `f` op, enzoverder, todat we de bitstring gedecodeerd hebben als `huffman`.

Opgave

In deze opgave vragen we je om strings te coderen naar bitstrings, en bitstrings te decoderen naar de oorspronkelijke string volgens het principe van de Huffmancodering. Hierbij worden bitstrings voorgesteld als strings die enkel bestaan uit de karakters `0` en `1`. We geven je de bitstrings die gebruikt worden om de verschillende symbolen te coderen onder de vorm van een tekstbestand, waarvan elke regel bestaat uit een symbool (één enkel karakter), gevolgd door een tab en de bitstring die gebruikt wordt om dit symbool voor te stellen. Hieronder zie je bijvoorbeeld de inhoud van het tekstbestand met de codering die hoort bij de boom uit de inleiding. Merk op dat het symbool op de eerste regel in dit voorbeeld een spatie is.

```
 111
a 010
e 000
f 1101
h 1010
i 1000
m 0111
n 0010
s 1011
t 0110
l 11001
o 00110
p 10011
r 11000
u 00111
x 10010
```

Definieer een klasse `Knoop` die kan gebruikt worden om de knopen van de binaire bomen voor te

stellen zoals die gebruikt worden bij Huffmancodering. Dit betekent dat elke knoop aan zijn linker- en/of rechtertak kan gelinkt zijn met een andere knoop (die we respectievelijk de linker- en rechterknoop noemen), en dat een knoop ook kan gelabeld zijn met een symbool (bladknopen hebben een symbool, interne knopen niet). De objecten van deze klasse moeten beschikken over methoden `links` en `rechts` die respectievelijk de linker- en rechterknoop teruggeven: zelf een object van de klasse `Knoop` indien er een linker- of rechterknoop is, of de waarde `None` indien er geen linker- of rechterknoop is. Daarnaast moeten de objecten van deze klasse ook beschikken over een methode `symbool` die het symbool teruggeeft waarmee de knoop gelabeld is: een karakter indien de knoop gelabeld is met een symbool of de waarde `None` indien de knoop niet gelabeld is met een symbool. Aan deze drie methoden moeten geen argumenten doorgegeven worden.

Definieer een klasse `Huffman` die kan gebruikt worden om strings te coderen als bitstrings, en bitstrings te decoderen naar de oorspronkelijke string volgens de Huffmancodering. Bij initialisatie van een object van deze klasse moet de locatie van een tekstbestand opgegeven worden dat de codering van de verschillende symbolen bevat in het formaat zoals hierboven omschreven. De initialisatiemethode moet ervoor zorgen dat het object een eigenschap `wortel` krijgt dat wijst naar de wortel van de binaire boom met de bitstringcodering van alle symbolen uit het gegeven bestand (een object van de klasse `Knoop`). Voorts moeten de objecten van deze klasse minstens de volgende methoden ondersteunen:

- Een methode `codeer` waaraan een string moet doorgegeven worden. De methode moet de Huffmangecodeerde bitstring teruggeven die correspondeert met de gegeven string. Indien de gegeven string een karakter bevat waarvan de bitstringcodering niet vervat zit in het bestand dat bij initialisatie werd opgegeven, dan moet de methode een `ValueError` opwerpen met de boodschap `onbekend symbool "x"`, waarbij `x` moet ingevuld worden met het eerste (meest linkse) karakter dat niet voorkomt in het bestand.
- Een methode `decodeer` waaraan een bitstring moet doorgegeven worden. De methode moet de oorspronkelijke string teruggeven die correspondeert met de gegeven string volgens het schema van de Huffmancodering. Indien de gegeven bitstring geen geldige stringvoorstelling heeft volgens het schema van de Huffmancodering, dan moet de methode een `ValueError` opwerpen met de boodschap `ongeldige bitstring`.

Voorbeeld

Bij onderstaande voorbeeldsessie gaan we ervan uit dat het tekstbestand codes.txt zich in de huidige directory bevindt.

```
>>> huffman = Huffman('codes.txt')
>>> isinstance(huffman.wortel, Knoop)
True
>>> huffman.wortel.rechts().rechts().rechts().symbool()
' '
>>> huffman.wortel.links().links().links().symbool()
'e'
>>> huffman.wortel.rechts().links().links().rechts().links().symbool()
'x'
>>> huffman.wortel.links().rechts().links().symbool()
'a'

>>> huffman.codeer('huffman')
'10100011111011101011101000010'
```

```
>>> len(huffman.codeer('huffman'))
28
>>> huffman.codeer('huffman codering')
Traceback (most recent call last):
ValueError: onbekend symbool "c"

>>> huffman.decodeer('10100011111011101011101000010')
'huffman'
>>> huffman.decodeer('10100011111011101011101000101')
Traceback (most recent call last):
ValueError: ongeldige bitstring
>>> huffman.decodeer('10100011111011101011010000010')
Traceback (most recent call last):
ValueError: ongeldige bitstring
```