

## DEFLATE

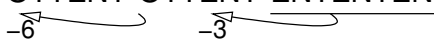
DEFLATE probeert twee methodes, letter-per-letter- en deelstringcodering, te combineren. In het slechtste geval is DEFLATE iets minder efficiënt dan Huffman codering (één of enkele bits meer in een bestand), als deelstringcodering nut heeft kan DEFLATE veel kortere resultaten geven.

In een eerste fase wordt onderzocht of deelstringcodering nuttig is. Als dit het geval is wordt deze uitgevoerd door middel van LZ77-codering, waarover later meer. In een tweede fase wordt dan onderzocht of Huffman codering nuttig is, en wordt deze eventueel uitgevoerd. Het resultaat is dat is dat DEFLATE bijna altijd een goede compressie oplevert, als dit tenminste mogelijk is. Het wordt dan ook toegepast bij algemene compressieprogramma's, zoals ZIP, GZIP, PKZIP, ZLIB en JAR. (Een volledige specificatie kan je terugvinden in RFC 1951.)

### Het basisprincipe van fase 1

LZ77 gebruikt de voorgaande tekst dient zelf als woordenboek. Het basisprincipe van deze vorm van compressie bestaat erin om deelstrings van de oorspronkelijke tekst die meer dan één keer voorkomen, vanaf de tweede keer voor te stellen als 'pointers' of 'indices' naar de eerste kopie van deze strings. Meer concreet: de gecomprimeerde uitvoerstream bestaat uit een reeks bytes en lengte/afstand-paren. Een lengte/afstand-paar verwijst naar een eerdere deelstring van de gegeven lengte die op een gegeven afstand (aantal bytes) vóór het huidige punt in het invoerbestand reeds is voorgekomen.

De string 'HOTTENTOTTENTENTENTOONSTELLING' kan bijvoorbeeld op de volgende manier worden gecomprimeerd :

origineel	H OTTENT OTTENT ENTENT ENTENT OONSTELLING
	
gecodeerd	<b>HOTTENT (6,6) (9,3) OONSTELLING</b>

(DEFLATE negeert herhalingen van minder dan 3 lettertekens.)

In de specificatie van DEFLATE wordt niet verteld hoe men stringherhalingen in de brontekst kan terugvinden, maar enkel hoe die moeten worden voorgesteld. Bij de klassieke standaard gaat men maximaal 32.768 plaatsen terug. Dit betekent dat, voor elke plaats in het bestand men de 32.768 plaatsen moet onderzoeken om te zien of men kan coderen. Dit is zeer rekenintensief, en men dient dus snelle zoekalgoritmen te gebruiken.

Soms is er keuze in de codering. Nemen we bijvoorbeeld de tekst

ananas, dialyse, analyse

die op twee manieren gecodeerd kan worden:

an(3,2)s, dialyse, (3,15)(4,9)

an(3,2)s, dialyse, an(5,9)

### Codering van fase 1

Natuurlijk kunnen we de voorstelling die we hierboven gebruikt hebben om het algoritme uit te leggen niet gebruiken om het bestand door te sturen. Het zou best kunnen dat het

oorspronkelijke bestand bijvoorbeeld de string “(3,2)” bevat, wat nogal verwarrend zou zijn.

De decodeerder weet, als hij klaar is met een deelbericht, niet of hij een bytekarakter of een lengte zal doorkrijgen: we moeten dus een alfabet maken dat zowel de gewone bytekarakters met EOSTREAM (in DEFLATE eigenlijk ENDOFBLOCK) bevat, samen met een aantal letters die aanduiden dat we met een lengte te doen hebben. In het totaal heeft dit alfabet 286 letters. Sommige van de lengtesymbolen worden nog gevolgd door enkele bittekens om de lengte verder te definiëren.

Vermits een afstand alleen na een lengte kan komen, en elke lengte gevolgd wordt door een afstand, kunnen we hier een ander alfabet gebruiken. Dit alfabet heeft 20 letters, maar sommige van deze letters worden nog gevolgd door extra bittekens.

Bytes worden voorgesteld met behulp van hun getalwaarde (0–255) en het getal 256 geeft het einde van de invoerstroom aan<sup>1</sup>. Voor lengtes worden getallen gebruikt tussen 257 en 285 zoals aangegeven in de tabel.

Lengte	Bits	Getal	Lengte	Bits	Getal	Lengte	Bits	Getal
3	0	257	15–16	1	267	67–82	4	277
4	0	258	17–18	1	268	83–98	4	278
5	0	259	19–22	2	269	99–114	4	279
6	0	260	23–26	2	270	115–130	4	280
7	0	261	27–30	2	271	131–162	5	281
8	0	262	31–34	2	272	163–194	5	282
9	0	263	35–42	3	273	195–226	5	283
10	0	264	43–50	3	274	227–257	5	284
11–12	1	265	51–58	3	275	258	0	285
13–14	1	266	59–66	3	276			

Voor grotere lengtes moeten enkele extra bittekens aan het getal worden toegevoegd (het aantal bevindt zich telkens in de tweede kolom). Zo wordt bijvoorbeeld lengte 239 voorgesteld met behulp van het getal 284 gevolgd door de bitstring 01100 — de binaire voorstelling van  $12 = 239 - 227$ . Andere voorbeelden: lengte 19 stel je voor als 269 gevolgd door 00, voor lengte 50 schrijf je 274 gevolgd door 111.

Voor afstanden gebruikt men een andere representatie die steunt op hetzelfde principe, met de volgende tabel.

Afstand	Bits	Getal	Afstand	Bits	Getal	Afstand	Bits	Getal
1	0	0	33–48	4	10	1025–1536	9	20
2	0	1	49–64	4	11	1537–2048	9	21
3	0	2	65–96	5	12	2048–3072	10	22
4	0	3	97–128	5	13	3073–4096	10	23
5–6	1	4	129–192	6	14	4097–6144	11	24
7–8	1	5	192–256	6	15	6145–8192	11	25
9–12	2	6	257–384	7	16	8193–12288	12	26
13–16	2	7	385–512	7	17	12289–16384	12	27
17–24	3	8	513–768	8	18	16385–24576	13	28
25–32	3	9	769–1024	8	19	24577–32768	13	29

Een afstand van 15 bytes wordt bijvoorbeeld voorgesteld als het getal 7 gevolgd door de bittekens 10. Merk op dat DEFLATE een maximum afstand toelaat van 32 kilobytes, en een maximale lengte van 258 bytes.

<sup>1</sup> In de praktijk zal DEFLATE de uitvoerstroom nog in verschillende blokken verdelen, en betekent 256 eigenlijk ‘einde van een blok’.

Met behulp van deze tabellen vinden we de volgende representatie voor de gecomprimeerde ‘HOTTENTOTTENTENTENTENTOONSTELLING’:

HOTTENT 260 4 1 263 2 OONSTELLING

## Tweede fase

Zoals reeds vermeld worden de getallen uit de eerste fase<sup>2</sup> later nogmaals vertaald met behulp van een prefixcode, waarbij de extra bittekens echter met rust worden gelaten. Hierbij voorziet DEFLATE twee verschillende mogelijkheden. (De gemaakte keuze wordt aangegeven met een speciale code in de hoofding van de gecomprimeerde uitvoerstream.)

Als eerste mogelijkheid wordt een *vaste* prefixcodering voorgesteld. Dit heeft het voordeel dat er geen tries moeten doorgestuurd worden. Afstandsgetallen worden gecodeerd als binaire getallen van 5 bittekens<sup>3</sup>. Byte- en lengtegetallen worden gecodeerd zoals in onderstaande tabel:

Getal	Bits	Code
0–143	8	00110000–10111111
144–255	9	110010000–111111111
256–279	7	0000000–0010111
280–287	8	11000000–11000111

(Bijvoorbeeld: 32 → 01000000, 160 → 110011000, 258 → 0000010.)

Als tweede mogelijkheid kan men een Huffman codering gebruiken voor de byte- en lengtegetallen enerzijds, en voor de afstandsgetallen anderzijds. De twee overeenkomstige Huffman bomen worden dan ook in de uitvoerstream doorgegeven.

De conclusie is dat DEFLATE zeer flexibel is. In fase 1 *kan* DEFLATE een codering met (lengte, verschuiving)-paren gebruiken, wat interessant is als stringcodering goede compressie levert. In fase 2 *kan* DEFLATE kiezen voor een vaste trie (die dan niet doorgestuurd moet worden) of voor een zelfberekende eigen trie. b

<sup>2</sup> De term ‘fase’ is hier een beetje ongelukkig gekozen: het is niet zo dat de volledige invoerstream eerst wordt omgezet naar getallen en pas daarna nogmaals wordt gecodeerd. In de praktijk gebeurt alles tegelijkertijd.

<sup>3</sup> Men vraagt zich dan af waarom men een alfabet van maar 30 letters heeft genomen in plaats van 32.