1. Inleiding

1.1. Wat is zippen?

Regelmatig moet je grote bestanden van de ene computer naar de andere doorgegeven. Dit doe je dan via het internet, via een netwerk, met diskettes of andere gegevensdragers. Ook wanneer je bestanden voor later wilt bewaren, zal het gemakkelijk zijn dat dit gebeurt in een zo compact mogelijke vorm.

Bestandscompressie is een techniek die je hiervoor kan gebruiken. In de volksmond wordt dit ook wel zippen genoemd. *Zippen* is eigenlijk een term die ontstaan is omdat veel mensen hetzelfde programma gebruikten om bestanden te comprimeren. Één van de eerste veel gebruikte toepassingen om bestanden te comprimeren was immers Pkzip. Dit programma produceert bestanden die de extensie *.zip* hebben. Op deze manier was de term zippen dus snel ingeburgerd.

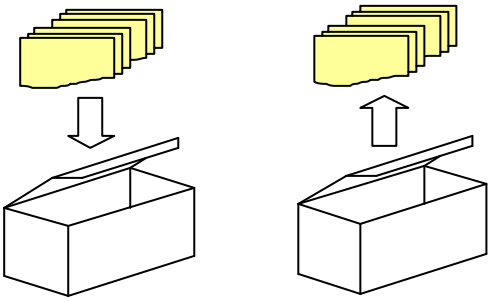
Bij zippen is het perfect mogelijk om verschillende bestanden samen te voegen (samen te pakken) in één archief. Zo kan je bijvoorbeeld gemakkelijk een groep bestanden verkleinen om samen door te sturen via E-mail.

Zippen is dus eigenlijk twee dingen:

- Samenpakken
- Comprimeren

1.1.1. Samenpakken

Samenpakken of inpakken van bestanden wil zeggen dat je er eigenlijk een doos rond zet. Je kan dit praktisch zeer goed vergelijken met een archiefdoos. Je hebt een aantal teksten die je wilt bewaren voor later. Deze steek je dan bij elkaar in een doos. Deze doos verhuisd dan naar het archief.

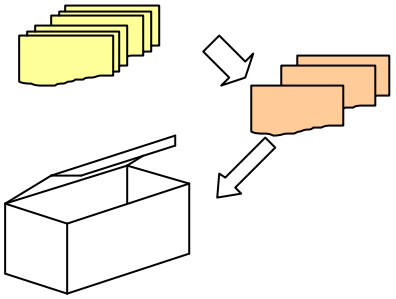


Samenpakken **Uitpakken**

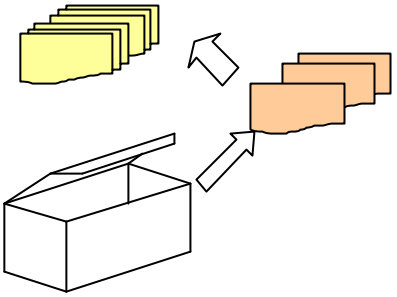
Op de computer gaat dit net zo. Je neemt een hoop bestanden bij elkaar om later gemakkelijk terug te kunnen vinden op een archief (een CD, een backup tape of een diskette).

1.1.2. Comprimeren

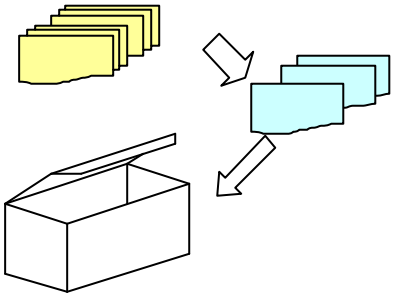
Comprimeren is het verkleinen van de grootte van een samengepakte groep bestanden. Er zijn twee vormen van comprimeren. Je kan een onderscheid maken tussen *Lossless* comprimeren en *Lossy* comprimeren. Lossless comprimeren wil zeggen dat je alles in de originele vorm terug uit de doos kan halen. Lossless comprimeren is eenvoudig gezegd verwijderen van redundante informatie. Hierover verder meer. Lossy comprimeren wil zeggen dat je de overbodig informatie gaat weggooien. Je kan bij Lossy comprimeren het origineel niet terugvinden. Dit wordt bijvoorbeeld toegepast voor comprimeren van figuren en geluid.



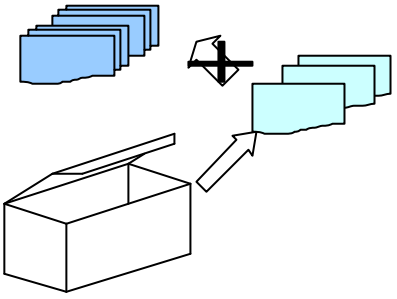
Comprimeren, redundante informatie verwijderen, lossless



Decomprimeren, Lossless, alles is terug uit het archief te halen



Comprimeren, overbodige informatie verwijderen, lossy



Decomprimeren, Lossy, niet alles is terug uit het archief te halen

Het programma Winzip past een Lossless compressie toe zodat je hetgeen je archiveert er ook terug kan uithalen. Winzip is niet het enige programma dat je kan gebruiken om Lossless te comprimeren. Een overzicht van andere compressieprogramma's vind je in de bijlage.

1.1.3. Techniek Lossless comprimeren

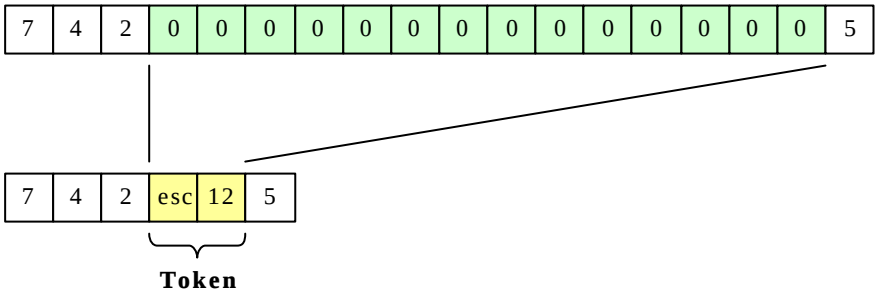
Deze cursus zou enorm complex worden wanneer alle compressietechnieken zouden worden bekeken. Dit is niet de bedoeling. Om de honger naar kennis te stillen en om een antwoord te geven op de vraag "Hoe doen ze het?" wordt hier kort uitgelegd hoe Lossless compressie kan

werken. Niet alle mogelijke technieken komen aan bod, maar het is een start. In de bijlage vind je meer gedetailleerde (Engelstalige) informatie over Lossless compressie.

1.1.3.1 Zero of blank onderdrukking

De eerste techniek die bij Lossless compressie gebruikt wordt is Zero of blank onderdrukking. Deze techniek is het eenvoudigst om te begrijpen. Wanneer één teken veel keer voorkomt in een reeks data kan je dit teken vervangen door een *token*. Een tokens is een teken of tekencombinatie die gebruikt kan worden in de plaats van repeterende tekens.

In onderstaand voorbeeld komt het getal 0 (zero) 14 keer achter elkaar voor in de reeks data. Dit getal kan vervangen worden door de token *esc 14*.



1.1.3.2 Run Length Encoding

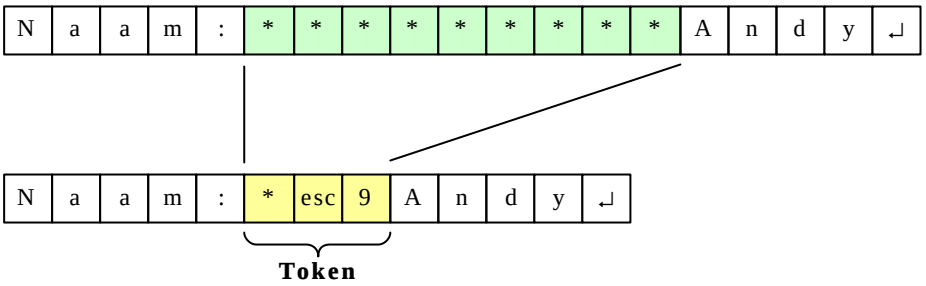
De tweede techniek die bij Lossless compressie gebruikt wordt is Run-Length Encoding. Deze techniek is zeer goed te vergelijken met de Zero of blank onderdrukking. Redundante data (verschillende keren achter elkaar hetzelfde teken) wordt vervangen door tokens.

Wanneer je bijvoorbeeld in een zin “Ik heb slaaaaaaaaaaaaaaaap.”de a’s zou vervangen door de token *a<esc><17>*, zou dit veel plaats besparen. Het teken *<esc>* geeft aan dat het vorige teken herhaald moet worden voor het aantal keer dat erna vermeld wordt (*<17>*).

Run Length Encoding toepassen op een gewone tekst heeft weinig zin omdat hierbij weinig redundante (herhalende) informatie voorkomt. Run Length Encoding wordt dan ook vaak gebruikt om figuren te comprimeren. In zwart-wit figuren (monochrome figuren) of figuren met weinig kleuren zit veel redundante data waardoor de compressie een zeer goed resultaat zal opleveren.

Het grote verschil met Zero of blank onderdrukking is dat eender welke reeks repeterende tekens vervangen kan worden door een token.

Run Length Encoding wordt ook wel toegepast om databases te comprimeren aangezien daar wel eens redundante of repeterende informatie in kan zitten.
In onderstaand voorbeeld komt het teken * 9 keer achter elkaar voor in de reeks data. Dit teken kan vervangen worden door de token ** esc 14*.

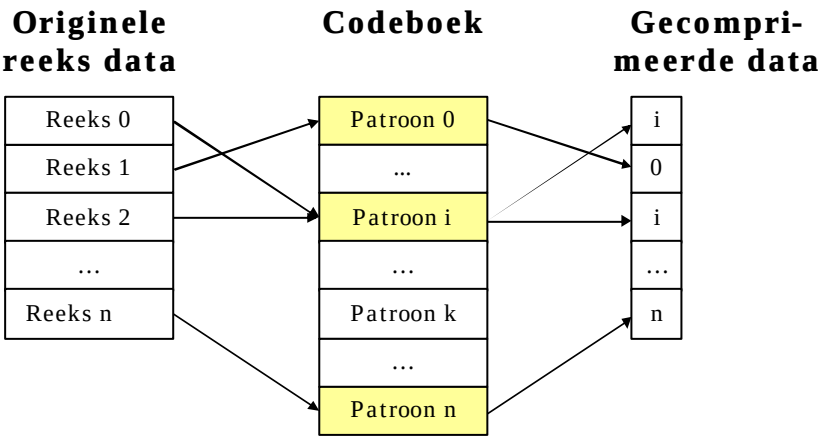


Bij het gebruiken van deze compressietechniek kunnen er een paar problemen opduiken. Één van de problemen is dat het *<esc>* teken kan voorkomen in de te comprimeren data. Vaak zal men dit oplossen door het *<esc>* teken te verdubbelen. Aangezien dit al snel zeer technisch wordt, wordt hier niet verder op ingegaan.

Een ander probleem is dat de lengte, die doorgaans opgegeven wordt na het te herhalen teken, niet groter kan zijn dan 256. Dit is de maximum lengte die met één byte kan opgegeven worden. De oplossing voor dit probleem is herhalen van de token.

1.1.3.3 Quantization Encoding

In de Quantization Encoding techniek wordt een codeboek van veel voorkomende patronen gemaakt. Dit codeboek wordt dan gebruikt om de datareeks te omschrijven. Onderstaand schema geeft aan wat er precies gedaan wordt.

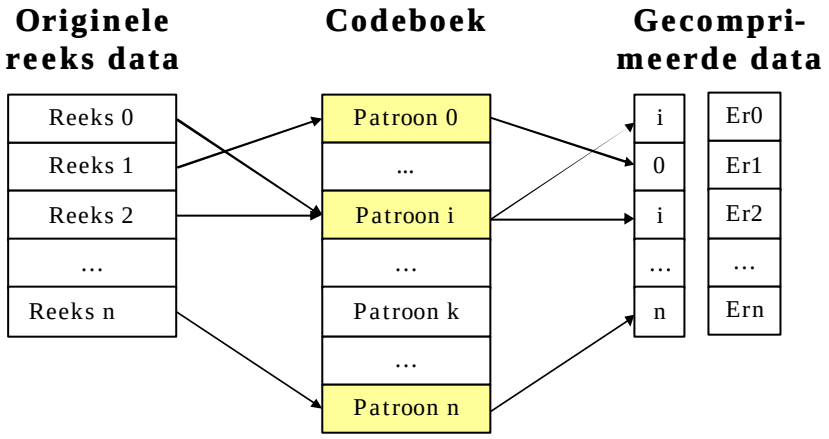


Wanneer de originele data bestaat uit verschillende los te definiëren patronen, kan de data in gecomprimeerde vorm geschreven worden als een opsomming van deze patronen. Reeks 0 van

de data voldoet bijvoorbeeld aan *patroon i*. Reeks 1 voldoet aan *patroon 1* en reeks 2 voldoet opnieuw aan *patroon i*. De data kan zo herschreven worden als *i0i*.

Wanneer er zeer veel verschillende patronen gedefinieerd moeten worden, of wanneer de patronen zeer kort zouden zijn, zal deze methode geen goede compressie opleveren. Het codeboek zal veel te groot worden. De plaats die bespaart wordt door gebruik van het codeboek wordt zo teniet gedaan. In zo'n geval kan men vaak beter een foutwaarde mee opnemen in de gecomprimeerde data.

In tegenstelling tot voorgaand geval voldoet Reeks 0 niet meer perfect aan patroon i. Er is een foutwaarde *Er0* te definiëren ten opzichte van het *patroon i*. Voor reeks 1, 2 en n valt op dezelfde manier te zeggen dat er een foutwaarde *Er1*, *Er2* en *Ern* te definiëren is ten opzichte van de respectievelijke benaderende patronen voor deze reeksen.



Door het opnemen van de foutwaarde zal het gecomprimeerde bestand groter worden. Goede compressiesoftware zal zelf bepalen wanneer het interessanter is om foutcodes te gebruiken in de plaats van een uitbreiding van het codeboek.

1.1.3.4 Hoffman coding

Bij deze efficiëntere Lossless compressietechniek wordt binaire codering gebruikt. De techniek zit moeilijker in elkaar dan de Run Length Encoding. Het basisidee achter de Hoffman coding is dat tekens die het meest voorkomen een korte binaire code krijgen. Tekens die het minst voorkomen krijgen een langere binaire code.

Het eerste wat je moet doen om een reeks data te comprimeren met behulp van de Hoffman coding is dus tellen hoeveel keer de tekens voorkomen in de reeks data.

De vraag die nu beantwoord moet worden is: welke binaire code moet aan welk teken gegeven worden om een zo ideaal mogelijke codering te bekomen?

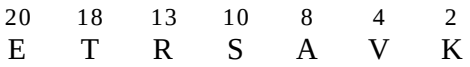
De regels om een Hoffman boom op te zetten zijn als volgt:

1. Zet alle tekens achter elkaar in dalende volgorde waarin ze voorkomen in de datareeks
2. Tel de twee kleinste waarden bij elkaar zolang deze som kleiner is dan de waarde juist groter
3. Herhaal stap 2 totdat je in de top bent
4. Plaats bij elk linkerbeen een 0, plaats bij elk rechterbeen een 1
5. Lees de code van de tekens af door vanuit de top naar elk teken te gaan

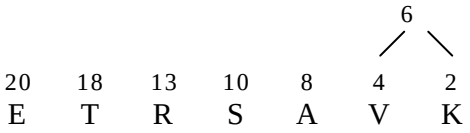
Neem bijvoorbeeld als resultaat van een telling het volgende:

E = 20, T = 18, R = 13, S = 10, A = 8, V = 4 en K = 2

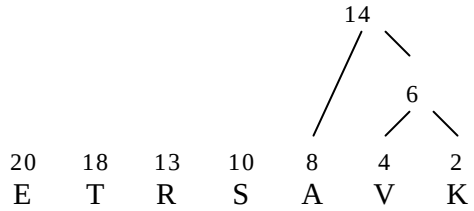
Stap 1:



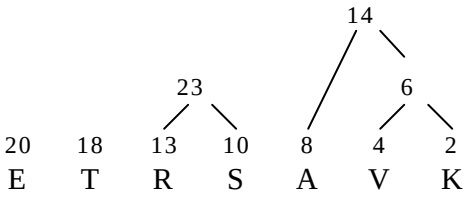
Stap 2 en 3:



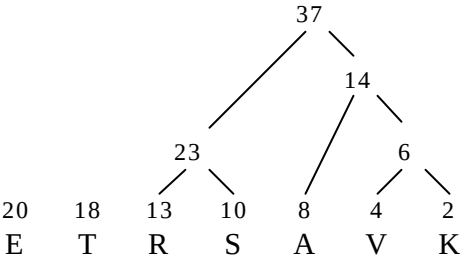
De 2 kleinste waarden optellen



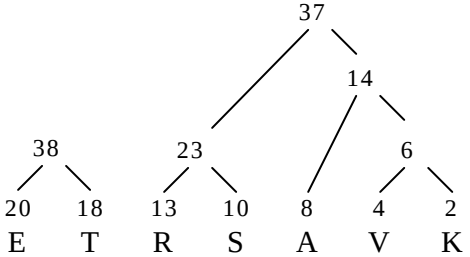
De 2 kleinste waarden optellen



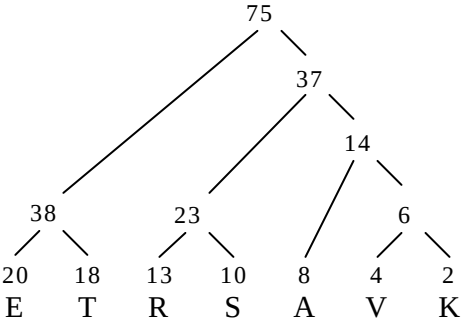
De 2 kleinste waarden optellen



De 2 kleinste waarden optellen

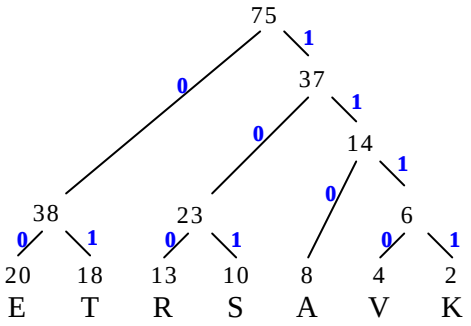


De 2 kleinste waarden optellen



De 2 kleinste waarden optellen

Stap 4:



De 2 kleinste waarden optellen

Stap 5:

E: 00, T: 01, R: 100, S: 101, A: 110, V: 1110, K: 1111

Het word REST coderen met de bovenstaande Hoffman boom geeft dus: 1000010101

Bij het decoderen van de binaire data is uiteraard de Hoffman boom vereist.

1.1.3.5 Rekenkundige (Arithmetic) coding

Rekenkundige coding lijkt zeer veel op de Hoffman coding, maar werkt meestal iets efficiënter. De basis voor rekenkundige coding is het maken van een schatting. Wanneer deze schatting juist is, is dit de meest effectieve manier van coderen. Wanneer de schatting verkeerd is, kan het zijn dat de gecomprimeerde data groter is dan het origineel. Dit is uiteraard niet de bedoeling.

1.1.4. Compressiegraad

De compressiegraad wordt bepaald door de grootte van de bestanden voor compressie te vergelijken met de grootte van het samengepakt bestand na het comprimeren. De compressiegraad wordt meestal uitgedrukt in procenten.

$$Compressiegraad = \frac{Groottena\ compressie}{Groottevoor\ compressie}$$

Wanneer een groep bestanden voor compressie 1.735 kiloBytes is en de samengepakte groep bestanden 334 kiloBytes groot is, dan is de compressiegraad van de groep bestanden 19,2 %.

Het kan zijn dat sommige van de individuele bestanden van de groep die samengepakt worden meer gecomprimeerd kunnen worden dan de andere. Je zou dus de compressiegraad van elk individueel bestand kunnen bekijken.

Wanneer 5 bestanden samen gecomprimeerd worden tot één bestand dan zouden de compressiegraden van alle afzonderlijke bestanden bijvoorbeeld zoals in onderstaande tabel kunnen zijn:

	Voor compressie	Na compressie	Compressiegraad
Bestand A	1.666 kB	1.550 kB	92,9 %
Bestand B	2.455 kB	1.045 kB	42,5 %
Bestand C	3.445 kB	1.209 kB	35,1 %
Bestand D	234 kB	198 kB	84,6 %
Bestand E	15.664 kB	3.447 kB	22,0 %
Totaal	23.464 kB	7.449 kB	31,7 %

1.2. Waar Winzip terugvinden?

Winzip is een programma dat in staat is om groepen bestanden te comprimeren. Naast dit programma bestaan er nog talloze programma's om groepen bestanden te comprimeren. Alternatieven zijn: WinRar, PKZip, ZipWave, FreeZip en TurboZip. Elk van deze programma's is eenvoudig terug te vinden op internet.

Het programma Winzip kan je downloaden op http://www.winzip.com/.