DCC–IME–USP

Routo Terada
Sala 288A Tel.: 818 6144
e-mail rt@ime.usp.br

MAC 172 – Técnicas de Construção de Algoritmos

Segundo semestre de 1997 - Exercício-programa 3

1. Algoritmo de compressão de dados LZ77

(J. Ziv and A. Lempel: A universal algorithm for data compression, IEEE Transactions on Information Theory, IT-23(3):337-343, May 1977.)

Variações deste algoritmo são implementadas e conhecidas como PKZip, Zip, LHarc, e ARJ.

A idéia básica do LZ77 é a de substituir uma seqüência S de caracteres por um código "apontando" para uma ocorrência anterior desta seqüência S. Obviamente quanto mais longa a seqüência substituida maior será a taxa de compressão atingida. Por exemplo se $S =$ "*abra*" como na figura a seguir, a segunda ocorrência de S é substituida por um tipo de "apontador" para a primeira ocorrência.

...	c	a	b	r	a	c	a	d	a	b	r	a	r	r	a	...

LZ77 analisa uma seqüência de caracteres (ou símbolos especiais codificados por exemplo em 8 bits) através de duas "janelas" contíguas que "deslizam" sobre a seqüência: uma janela chamada *Janela de Busca* que contém os últimos B caracteres já comprimidos, e a outra janela chamada *Janela Adiante* contendo os próximos A caracteres que deverão ser comprimidos ainda. B e A são *parâmetros variáveis* do algoritmo. Na figura a seguir, a Janela de Busca contém $B = 8$ caracteres: "*eabcaead*", e a Janela Adiante contém $A = 7$ caracteres: "abcacca". Na prática os parâmetros B e A são muito maiores do que neste exemplo, mas por simplicidade didática vamos usar valores pequenos.

aponta
↓

...	e	a	b	c	a	e	a	d	a	b	c	a	c	c	a	c ...

Janela de Busca (já compactada) 8 caracteres — Janela Adiante (ainda não compact.) 7 caracteres

Para comprimir alguma subseqüência contida na Janela Adiante, LZ77 desloca um apontador desde a posição do caractere *mais à direita* da Janela de Busca para a esquerda, até encontrar um caractere igual ao *primeiro caractere* na Janela Adiante: neste exemplo, é o caractere “a''. A distância do apontador até o início da Janela Adiante é chamado de *deslocamento d*. Neste exemplo $d = 7$. LZ77 examina os caracteres seguintes ao apontado para verificar se são iguais aos caracteres consecutivos na Janela Adiante. O número de caracteres iguais nesta verificação é chamado de *comprimento de igualdade c*. Neste exemplo, $c = 4$. LZ77 procura a igualdade mais longa possível, é claro. Depois de achar a igualdade mais longa, LZ77 emite a saída $< d, c, C(s) >$, onde d é o deslocamento, c é o comprimento de igualdade e $C(s)$ é o código do caractere na Janela Adiante logo seguinte à igualdade. Neste exemplo a saída é $< d = 7, c = 4, C(“c”) >$, pois “c'' é o caractere seguinte a “$abca''$ na Janela Adiante. Se nenhum caractere na Janela de Busca for igual ao primeiro caractere na Janela Adiante, então a saída será $< 0, 0, C(s) >$, isto é, $d = 0$ e $c = 0$ no triplo indica que a compressão não foi possível e o próprio código do caractere é a saída.

Sendo B o comprimento da Janela de Busca e sendo A o comprimento da Janela Adiante, e sendo C o tamanho do conjunto de caracteres possíveis (i.e., o alfabeto), o número de “bits” necessários para codificar os triplos $< d, c, C(s) >$ usando *códigos de comprimento fixo*, é $\lceil \log B \rceil + \lceil \log(B + A) \rceil + \lceil \log C \rceil$. Note que o segundo termo é $\lceil \log(B + A) \rceil$ e não $\lceil \log(A) \rceil$ pois veremos mais adiante que a igualdade pode em certos casos ultrapassar o fim da Janela de Busca.

Nos exemplos a seguir ilustramos as três situações que podem ocorrer durante o processo de compressão:

1. Não há igualdade na Janela de Busca para o primeiro caractere a ser comprimido na Janela Adiante.

2. Há igualdade.

3. A igualdade se estende até a Janela Adiante, além do extremo direito da Janela de Busca.

Vamos supor que temos a seqüência: “...*cabraca*dabrarrarrad...” e seja $B = 7$ o comprimento da Janela de Busca contendo “cabraca”, e seja $A = 6$ o comprimento da Janela Adiante contendo “dabrar”.

...	c	a	b	r	a	c	a	d	a	b	r	a	r	r ...

Situação 1: Não há igualdade na Janela de Busca para o primeiro caractere a ser comprimido na Janela Adiante.

Note que não há caractere "d'' na Janela de Busca. Então a saída do LZ77 é $< 0, 0, C(\text{“}d\text{”}) >$, onde o par $(0, 0)$ indica a não ocorrência de "*d*", e $C(\text{“}d\text{”})$ é o código de "*d*". Após avançar uma posição para a direita, devido ao "*d*", temos a seguinte situação:

...	c	a	b	r	a	c	a	d	a	b	r	a	r	r	a	...

Situação 2: Há igualdade na Janela de Busca para o primeiro caractere a ser comprimido na Janela Adiante.

...	a	b	r	a	c	a	d	a	b	r	a	r	r	a	...

Analisando a Janela de Busca "abracad" da direita para a esquerda, notamos o primeiro "a'' com deslocamento $d = 2$; o comprimento de igualdade $c = 1$. Examinando mais à esquerda, LZ77 encontra uma segunda igualdade com deslocamento $d = 4$, e comprimento $c = 1$ de novo. Examinando mais à esquerda ainda, LZ77 encontra uma terceira igualdade com deslocamento $d = 7$, e comprimento $c = 4$, que é a mais longa. Então, a saída será o triplo $< 7, 4, C(\text{“}r\text{”}) >$, e deslocamos as janelas de *cinco* (e não 4; por quê?) posições.

aponta

↓ $d = 7$

...	c	a	b	r	a	c	a	d	a	b	r	a	r	r	...

Janela de Busca — Janela Adiante

$c = 4$ — (6 cars.)

Após o deslocamento das janelas de cinco posições para a direita obtemos:

...	a	d	a	b	r	a	r	r	a	r	r	a	d	...

Situação 3: A igualdade se estende até a Janela Adiante, além do extremo direito da Janela de Busca.

A Janela Adiante contém "rarrad". Há uma igualdade para "r'' na Janela de Busca com deslocamento $d = 1$, e comprimento $c = 1$, e uma segunda igualdade com deslocamento $d = 3$ e comprimento aparente $c = 3$. Na realidade podemos utilizar comprimento $c = 5$ no lugar de 3, devido a "$rarra''$. A razão desta possibilidade ficará clara logo a seguir quando veremos o processo de descompressão.

2. Algoritmo LZ77 em processo de descompressão

Vamos supor que LZ77 já descomprimiu os códigos correspondentes a "$cabraca''$, e LZ77 tem a seguir os códigos:

$$< 0, 0, C(\text{“}d\text{”}) >, < 7, 4, C(\text{“}r\text{”}) >, \text{ e } < 3, 5, C(\text{“}d\text{”}) >.$$

Vamos ver como descomprimir $< 0, 0, C(\text{“}d\text{”}) >$ e $< 7, 4, C(\text{“}r\text{”}) >$, obtendo:

...	c	a	b	r	a	c	a	d	a	b	r	a	r	r	...

O código $< 0, 0, C(\text{“}d\text{”}) >$ é fácil de descomprimir obtendo-se "cabracad":

A seguir a decodificação de $< 7, 4, C(\text{“}r\text{”}) >$. O valor 7 no triplo diz ao LZ77 "desloque 7 posições para a esquerda na Janela de Busca", e o valor 4 diz "copie 4 caracteres a partir deste ponto" para a saída. Uma ilustração destes passos está a seguir.

aponta 7 p/ esqu.

↓

c	a	b	r	a	c	a	d

copia 1

↓ → ↓

c	a	b	r	a	c	a	d	a

copia 2

↓ → ↓

c	a	b	r	a	c	a	d	a	b

copia 3

$\downarrow \quad \rightarrow \quad \downarrow$

c	a	b	r	a	c	a	d	a	b	r

copia 4

$\downarrow \quad \rightarrow \quad \downarrow$

c	a	b	r	a	c	a	d	a	b	r	a

E depois anexe "r" na saída, devido a $C($"r"$)$ e desloca as janelas de cinco posições.

decodifica "r''

$\downarrow$

c	a	b	r	a	c	a	d	a	b	r	a	r

A seguir, finalmente LZ77 examinará e descomprimirá o triplo $< 3, 5, C($"d"$) >$ para obtermos a seguinte situação:

...	a	d	a	b	r	a	r	r	a	r	r	a	d	...

aponta 3

$\downarrow$ p/ esq

...	a	d	a	b	r	a	r

copia 1

$\downarrow \quad \rightarrow \quad \downarrow$

...	a	d	a	b	r	a	r	r

copia 2

$\downarrow \quad \rightarrow \quad \downarrow$

...	a	d	a	b	r	a	r	r	a

copia 3

$\downarrow \quad \rightarrow \quad \downarrow$

...	a	d	a	b	r	a	r	r	a	r

copia 4

$\downarrow \quad \rightarrow \quad \downarrow$

...	a	d	a	b	r	a	r	r	a	r	r

copia 5

$\downarrow \quad \rightarrow \quad \downarrow$

...	a	d	a	b	r	a	r	r a r r a

decodi-
fica "d''" $\downarrow$

a	d	a	b	r	a	r	r a r r a d

3. Conclusões

LZ77 é um algoritmo simples do tipo *adaptativo*, pois adapta-se às seqüências que vão sendo processadas, sem qualquer hipótese necessária sobre a freqüência de cada caractere. Constrói um *dicionário* com as subseqüências da Janela de Busca, à medida que a compressão vai sendo efetuada. O leitor pode perceber facilmente que a taxa de compressão é melhor quanto maior for a ocorrência de padrões iguais próximas entre si.

As variações do LZ77 consistem em projetar diversas formas eficientes de se codificar os triplos $< d, c, C(s) >$ que o algoritmo gera.

1. Uma modificação é utilizar códigos de comprimento variável para d e c (ao contrário de se utilizar códigos de comprimento fixo como estávamos supondo anteriormente).

2. Uma outra codificação bem simples objetiva eliminar a necessidade de se usar um triplo da forma $< 0, 0, C(s) >$ para representar apenas um caractere: o uso de um triplo é muito ineficiente especialmente se um grande número de caracteres ocorre infrequentemente (por quê?). A modificação para se evitar tal ineficiência é a simples inclusão de um "flag" f que só pode ser verdadeiro ou falso. A inclusão de f torna desnecessário o terceiro elemento de cada triplo: o algoritmo passa a gerar ou $< f = falso, d, c >$ ou $< f = verdadeiro, C(s) >$. Esta modificação do LZ77 é conhecida por LZSS (J.A. Storer and T.G. Syzmanski: Data compression via textual substitution, Journal of the ACM, 29:928-951, 1982).

4. Exercício-programa

Implemente em Java o algoritmo LZ77 com as modificações (1) e (2) descritas acima, para comprimir e descomprimir arquivo em disco, do tipo *texto ou binário*

(.exe). Uma possibilidade é leitura de 8 em 8 bits. O programa deve ter interface gráfica onde pergunta, no mínimo:

1. o nome do arquivo de entrada e o de saída.
2. os parâmetros B (comprimento da Janela de Busca) e A (da Janela Adiante).

Ademais, o programa deve informar no final qual a taxa de compressão obtida, em porcentagem.

Opcionalmente, quando é para comprimir um arquivo, o programa poderia aumentar automaticamente B e A de uma certa maneira (que V pode projetar e fixar) e comprimir de novo *se* uma certa taxa de compressão T *não* for atingida para aquele arquivo. V pode fixar T ou perguntar ao usuário no início. Os programas comerciais chegam a comprimir para menos de 60 porcento no caso de arquivos com texto.

Elabore e documente *profissionalmente* o seu programa. V pode concorrer a prêmios no próximo concurso de software SOFTEST do DCC-USP e enriquecer o seu currículo!

5. Prazo para entrega

Dia 20 de novembro de 1997.