

Huffman coding:

O algoritmo de Huffman é um algoritmo de compressão de texto 'lossless', ou seja, que não perde informação no processo de compressão. (veja também 'lossy compression' **Sayood, Khalid.**

2018. *Introduction to Data Compression*. 5. s.l. : Morgan Kaufmann, 2018. pp. 221-255.)

Partindo do princípio de que queremos armazenar e transmitir texto, o algoritmo de Huffman oferece uma maneira mais eficiente de representar os bits do texto de forma não ambígua. [1]

Problema:

Para representarmos texto em um computador com 0s e 1s, precisamos inicialmente definir como interpretar essa 'string' de bits. A maneira mais conhecida é usando uma tabela como a tabela ASCII.

The ASCII code

American Standard Code for Information Interchange

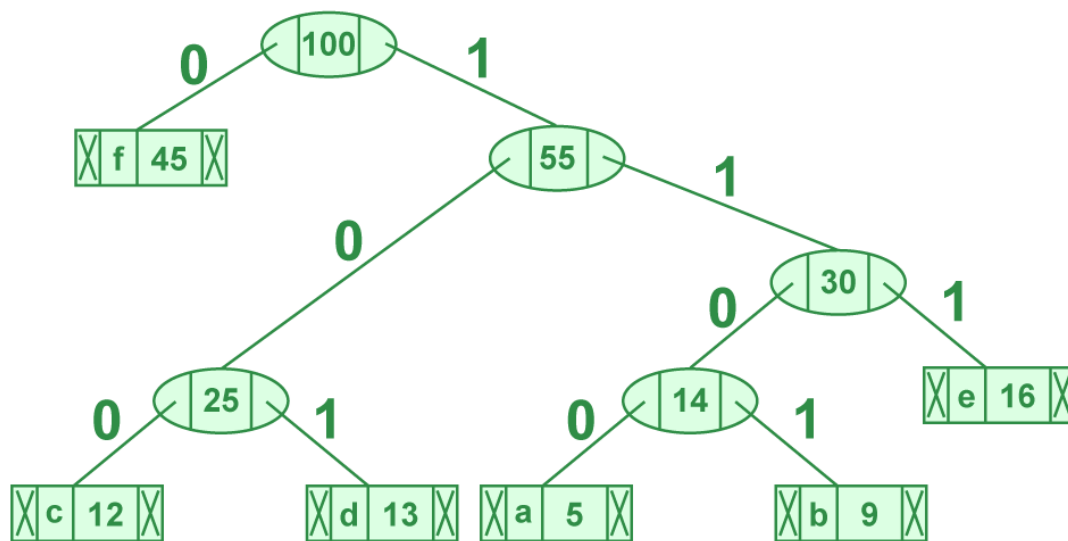
ASCII control characters			ASCII printable characters						Extended ASCII characters														
DEC	HEX	Símbolo ASCII	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo			
00	00h	NULL (carácter nulo)	32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	à	192	C0h	Ł	224	E0h	Ó
01	01h	SOH (inicio encabezado)	33	21h	!	65	41h	A	97	61h	a	129	81h	Û	161	A1h	á	193	C1h	ł	225	E1h	ô
02	02h	STX (inicio texto)	34	22h	"	66	42h	B	98	62h	b	130	82h	Ü	162	A2h	â	194	C2h	Ł	226	E2h	Ô
03	03h	ETX (fin de texto)	35	23h	#	67	43h	C	99	63h	c	131	83h	Ý	163	A3h	ã	195	C3h	ł	227	E3h	Õ
04	04h	EOT (fin transmisión)	36	24h	\$	68	44h	D	100	64h	d	132	84h	ÿ	164	A4h	ä	196	C4h	Ł	228	E4h	Ö
05	05h	ENQ (enquiry)	37	25h	%	69	45h	E	101	65h	e	133	85h	ÿ	165	A5h	Å	197	C5h	ł	229	E5h	Ø
06	06h	ACK (acknowledgement)	38	26h	&	70	46h	F	102	66h	f	134	86h	ÿ	166	A6h	Ä	198	C6h	Ł	230	E6h	Ù
07	07h	BEL (timbre)	39	27h	'	71	47h	G	103	67h	g	135	87h	ÿ	167	A7h	Å	199	C7h	ł	231	E7h	Ú
08	08h	BS (retroceso)	40	28h	(72	48h	H	104	68h	h	136	88h	ÿ	168	A8h	Ä	200	C8h	Ł	232	E8h	Û
09	09h	HT (tab horizontal)	41	29h)	73	49h	I	105	69h	i	137	89h	ÿ	169	A9h	Å	201	C9h	ł	233	E9h	Ü
10	0Ah	LF (salto de línea)	42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	ÿ	170	AAh	Ä	202	CAh	Ł	234	EAh	Ý
11	0Bh	VT (tab vertical)	43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ÿ	171	ABh	Å	203	CBh	ł	235	EBh	ÿ
12	0Ch	FF (form feed)	44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	ÿ	172	ACH	Å	204	CCh	Ł	236	ECh	ü
13	0Dh	CR (retorno de carro)	45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ÿ	173	ADh	Å	205	CDh	ł	237	EDh	ÿ
14	0Eh	SO (shift Out)	46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	ÿ	174	Aeh	Å	206	CEh	Ł	238	Eeh	ÿ
15	0Fh	SI (shift in)	47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	ÿ	175	Afh	Å	207	CFh	ł	239	Efh	ÿ
16	10h	DLE (data link escape)	48	30h	0	80	50h	P	112	70h	p	144	90h	ÿ	176	B0h	Å	208	D0h	Ł	240	F0h	ÿ
17	11h	DC1 (device control 1)	49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	Å	209	D1h	ł	241	F1h	±
18	12h	DC2 (device control 2)	50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	Å	210	D2h	Ł	242	F2h	¼
19	13h	DC3 (device control 3)	51	33h	3	83	53h	S	115	73h	s	147	93h	ø	179	B3h	Å	211	D3h	ł	243	F3h	½
20	14h	DC4 (device control 4)	52	34h	4	84	54h	T	116	74h	t	148	94h	ø	180	B4h	Å	212	D4h	Ł	244	F4h	¾
21	15h	NAK (negative acknowledge)	53	35h	5	85	55h	U	117	75h	u	149	95h	ø	181	B5h	Å	213	D5h	ł	245	F5h	÷
22	16h	SYN (synchronous idle)	54	36h	6	86	56h	V	118	76h	v	150	96h	ù	182	B6h	Å	214	D6h	Ł	246	F6h	×
23	17h	ETB (end of trans. block)	55	37h	7	87	57h	W	119	77h	w	151	97h	ÿ	183	B7h	Å	215	D7h	ł	247	F7h	÷
24	18h	CAN (cancel)	56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	Å	216	D8h	Ł	248	F8h	÷
25	19h	EM (end of medium)	57	39h	9	89	59h	Y	121	79h	y	153	99h	ÿ	185	B9h	Å	217	D9h	ł	249	F9h	÷
26	1Ah	SUB (substitute)	58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	ÿ	186	BAh	Å	218	DAh	Ł	250	FAh	÷
27	1Bh	ESC (escape)	59	3Bh	;	91	5Bh	[123	7Bh	{	155	9Bh	ø	187	BBh	Å	219	DBh	ł	251	FBh	÷
28	1Ch	FS (file separator)	60	3Ch	<	92	5Ch	\	124	7Ch	}	156	9Ch	ø	188	BCh	Å	220	DBh	Ł	252	FBh	÷
29	1Dh	GS (group separator)	61	3Dh	=	93	5Dh]	125	7Dh	~	157	9Dh	ø	189	BDh	Å	221	DDh	ł	253	FBh	÷
30	1Eh	RS (record separator)	62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	x	190	BEh	Å	222	DEh	Ł	254	FEh	÷
31	1Fh	US (unit separator)	63	3Fh	?	95	5Fh	_				159	9Fh	f	191	BFh	Å	223	DFh	ł	255	FFh	÷
127	20h	DEL (delete)																					

Esta representação usa 8 bits para cada caractere da tabela. Porém, em um texto normal, além de repetirmos muitas letras, não usamos todos os caracteres da tabela ASCII.

Estratégia:

A estratégia primária do algoritmo de Huffman é construir uma tabela nova com base na frequência dos caracteres usados no texto. Dado que a letra 'x' aparece com maior frequência, queremos representar 'x' com o menor número de bits possível.

Dessa forma, podemos representar o texto com uma [árvore binária](#).



Mantendo todos os caracteres como nós folhas, não haverá ambiguidade, pois saberemos que, ao chegar em um caractere após seguir uma `string` de bits, podemos parar e ir para o próximo caractere.

A primeira coisa a ser feita no algoritmo é montar uma tabela de frequência, mapeando cada caractere com sua respectiva frequência no texto.

Depois, construímos uma fila de prioridade com base nessa frequência para a construção da árvore, pois queremos que o caractere com maior frequência seja montado mais próximo à raiz da árvore (fazendo com que ele tenha menos bits de representação). Com a fila montada de maneira que a "menor frequência sai primeiro", construímos a árvore da seguinte forma:

```
while queue.len > 1
    c1 = queue.pop()
    c2 = queue.pop()
    root = node {left: c1, right: c2}
    root.freq = c1.freq + c2.freq
    queue.insert(root)
return queue.pop()
```

Pegue os dois primeiros nós da fila;

Monte uma sub-árvore com root tendo frequência igual a soma desses dois nós;

Insira esse novo nó, agora root, na fila de prioridade;

Repita o processo até sobrar apenas o root;

Dessa forma, resolvemos o problema do número fixo de bits da tabela, já que todos os nós agora são folhas, devido à criação dos nós nulos com frequência média no meio da árvore. Assim, dado qualquer binário, é possível navegar pela árvore e recuperar o texto original a partir de qualquer sequência binária, garantindo que nenhuma informação seja perdida.

References

Sayood, Khalid. 2018. *Introduction to Data Compression*. 5. s.l. : Morgan Kaufmann, 2018.