ハフマン符号化のプログラム演習

ハフマン符号化復号化を行うプログラムの作成を目的とする．奥村晴彦著「c言語によるアルゴリズム事典」を参考にしたが，ヒープを使っている部分などを理解を容易にするため使わないようにデチューンした．そのため速度はオリジナルの奥村先生のものより遅くなっています．

課題　適当なファイルを　a.out　e　ファイル名　符号化したファイル名

でハフマン符号化 し，　a.out　d　符号化したファイル名　ファイル名　で復号化するような

プログラムを作成せよ．

まずhuffman.c，bitio.c の2つのファイルを各自のディレクトリにコピーしてもらいたい（ブラウザでダウンロードする）．

1ビットをファイルに書き込むにはputbit()を用いる．また，1ビットをファイルから読み込むにはgetbit()を用いる．これらの関数を用いるには#include “bitio.c”とする．main()の部分は次のように作れば良いだろう．

```
#include "bitio.c"            /* ビット入出力 */
#define N       256           /* 文字の種類 */
#define CHARBITS  8           /* 1バイトのビット数 */
#define MIN_INIT 1000000
int parent[2*N-1], left[2*N-1], right[2*N-1];  /* Huffman木 */
int Y_N[2*N-1];                                /* ハフマン木をつくるとき調べようとしている語を，
                                                  すでに調べた=1，まだ調べていない=0   */
unsigned long int freq[2*N-1];  /* 各文字の出現頻度 */
/* プロトタイプ宣言 */
int choose(void);
void encode(void);
void decode(unsigned long int);
```

```
int main(int argc, char *argv[])
{
        int c;
        unsigned long int size;  /* 元のバイト数 */

        if (argc != 4 || ((c = *argv[1]) != 'E' && c != 'e'
                        && c != 'D' && c != 'd'))
                error("使用法は本文を参照してください");
        if ((infile  = fopen(argv[2], "rb")) == NULL)
                error("入力ファイルが開きません");
        if ((outfile = fopen(argv[3], "wb")) == NULL)
                error("出力ファイルが開きません");
        if (c == 'E' || c == 'e') {
                fseek(infile, 0L, SEEK_END);  /* infile の末尾を探す */
                size = ftell(infile);     /* infile のバイト数 */
                fwrite(&size,sizeof size,1,outfile);
                rewind(infile);
                encode();  /* 圧縮 */
        } else {
                fread(&size, sizeof size, 1, infile);  /* 元のバイト数 */
                decode(size);  /* 復元 */
        }
        fclose(infile); fclose(outfile);
                return EXIT_SUCCESS;
}
```

入力ファイルはinfile出力ファイルはoutfileとして扱うことになる.

入力ファイルのサイズをunsigned long int ＝4バイト？で出力ファイルに書き込んでおく. 出力ファイルはバイナリでEOFが無いため復号化のとき必要になる.

復号化のとき，まずunsigned long int型のsizeという変数にもとの入力ファイルのサイズを代入しておく.

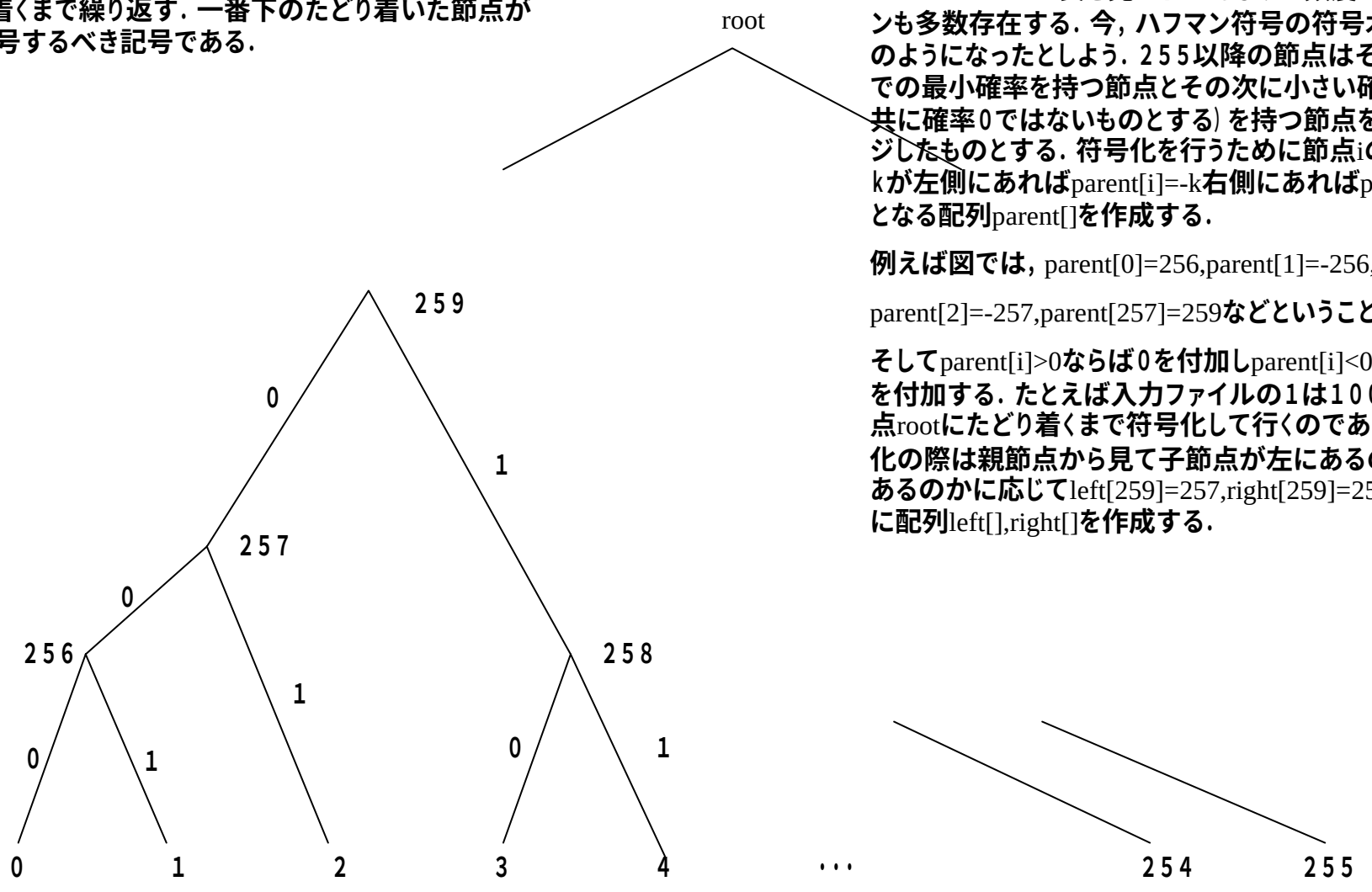
頂点rootから出発し，decodeするべきパターンが0ならば節点left[root]に移動し，1だったらright[root]に移動する．これを一番下の節点0～255にたどり着くまで繰り返す．一番下のたどり着いた節点が復号するべき記号である．

情報源アルファベットとしては8ビットの全パターンを考える．入力ファイルがテキストファイルならば次のページのASCIIコード表を見てわかるように頻度0のパターンも多数存在する．今，ハフマン符号の符号木が下図のようになったとしよう．255以降の節点はその時点での最小確率を持つ節点とその次に小さい確率（但し共に確率0ではないものとする）を持つ節点をマージしたものとする．符号化を行うために節点iの親節点kが左側にあればparent[i]=-k右側にあればparent[i]=kとなる配列parent[]を作成する．

例えば図では，parent[0]=256,parent[1]=-256,

parent[2]=-257,parent[257]=259などということになる．

そしてparent[i]>0ならば0を付加しparent[i]<0ならば1を付加する．たとえば入力ファイルの1は100･･･と節点rootにたどり着くまで符号化して行くのである．復号化の際は親節点から見て子節点が左にあるのか右にあるのかに応じてleft[259]=257,right[259]=258のように配列left[],right[]を作成する．



下位4ビット＼上位3ビット	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	”	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	＼	l	|
D	CR	GS	−	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	／	?	O	_	o	DEL

アスキーコード

関数encode()の構成

(1) 符合化したビットを表す配列codebit[8]をchar型で用意しておく.

(2) freq[i] = 0, Y_N[i] = 1として 頻度等の初期化をする. freq[i]は各パターンの頻度, Y_N[i]=0, まだその節点をマージしていない, Y_N[i]=1, すでにその節点はマージされている, をあらわすものとする.

(3) 入力ファイルのファイルエンドまで調べ freq[i]およびY_N[i]の更新をせよ. freq[i]>0ならばY_N[i]＝0としておく.

(4)

```
for(i=0;i<N;i++){
   fwrite(&freq[i],sizeof(unsigned long int),1,outfile);
 }
```

として (4) で求めた頻度を出力ファイルに書き込んでおく. decode()を見るとこの情報を利用してfreq[]の復元を行っているのがわかるだろう.

(5) while((i=choose())!=-1&&(j=choose())!=-1){ }として最小の確率を持つ節点iとその次に小さい確率を持つ節点jをとってくる. {}内では親節点kの頻度freq[k]およびparent[i],parent[j]の更新を行う.

(6) rewind(infile)を行い入力ファイルの先頭にファイルポインタinfileを戻す.

(7) 入力ファイルの文字を1バイトずつ調べ, codebit[k] = 0または1を決定せよ. root節点まで戻れたかどうかはparent[j]=0かどうかで判定できる.

(8) 決定できたらwhile (--k >= 0) putbit(codebit[k]);として出力ファイルに1ビットずつ書き込め.

(9) putbits(7, 0); /* バッファの残りをフラッシュ */

以上