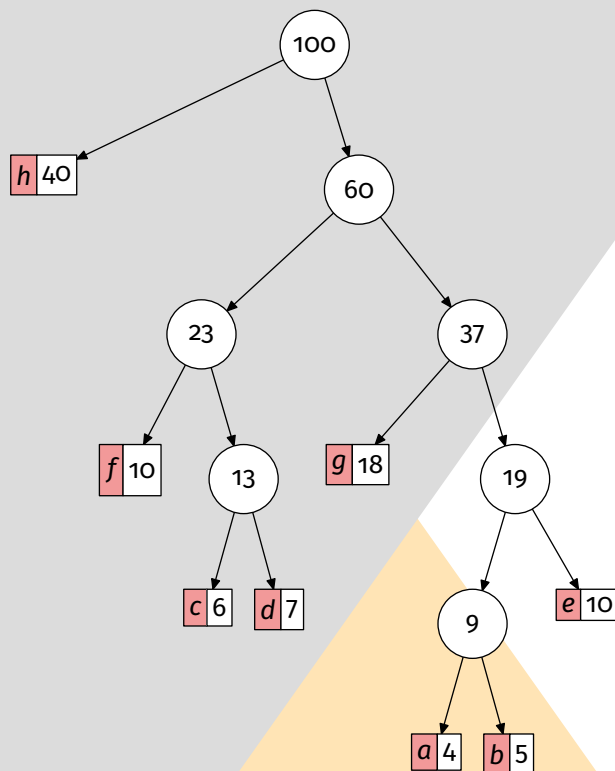


# huffman

drawing binary Huffman trees  
with METAPOST and METAOBJ



**Contributor**  
Maxime CHUPIN  
[notezik@gmail.com](mailto:notezik@gmail.com)

Version 0.1, 2023, April, 21th  
<https://plmlab.math.cnrs.fr/mchupin/huffman>

## Abstract

This METAPOST package allows to draw binary Huffman trees from two arrays : a string one, and a value one. It is based on METAOBJ package which provides many tools to build trees in general.

<https://plmlab.math.cnrs.fr/mchupin/huffman>  
<https://github.com/chupinmaxime/huffman>

## Contents

<b>1</b>	<b>Installation</b>	<b>2</b>
1.1	With T <sub>E</sub> Xlive under Linux or macOS . . . . .	2
1.2	With MikT <sub>E</sub> X and Windows . . . . .	3
1.3	Dependencies . . . . .	3
<b>2</b>	<b>Main Command</b>	<b>3</b>
<b>3</b>	<b>Package Options</b>	<b>4</b>
<b>4</b>	<b>METAOBJ Tree Options</b>	<b>6</b>
<b>5</b>	<b>Nodes and Subtrees Access</b>	<b>7</b>
5.1	Tree Number . . . . .	7
5.2	Access to Nodes and Leaves . . . . .	7
<b>6</b>	<b>Constructors</b>	<b>9</b>

*This package is in beta version—do not hesitate to report bugs, as well as requests for improvement.*

## 1 Installation

huffman is on CTAN and can also be installed via the package manager of your distribution.

<https://www.ctan.org/pkg/huffman>

### 1.1 With T<sub>E</sub>Xlive under Linux or macOS

To install huffman with T<sub>E</sub>XLive, you will have to create the directory texmf directory in your home.

```
user $> mkdir ~/texmf
```

Then, you will have to place the huffman.mp file in the

~/texmf/metapost/huffman/

Once this is done, `huffman` will be loaded with the classic METAPOSTinput code

```
input huffman
```

## 1.2 With MikTeX and Windows

These two systems are unknown to the author of `huffman`, so we refer you to the MikTeX documentation concerning the addition of local packages:

<http://docs.miktex.org/manual/localadditions.html>

## 1.3 Dependencies

`huffman` depends, of course on METAPOST [2], as well as the packages `metaobj` [1] and—if `huffman` is not used with LuaTeX and the `luamplib` package—the `latexmp` package.

## 2 Main Command

The package `huffman` provides one principal command (which is a METAOBJ like constructor):

```
newBinHuffmanTree.<name>(<sizeofarrays>)(<symparray>,<valuearray>)
```

**<name>**: is the name of the object;

**<sizeofarray>**: is the size (integer) of the arrays;

**<symparray>**: is the array of **string** containing the symboles;

**<valuearray>**: is the array of **numeric** containing the values associated to the symboles.

The data arrays should begin at index 1.

### Exemple 1

```
input huffman

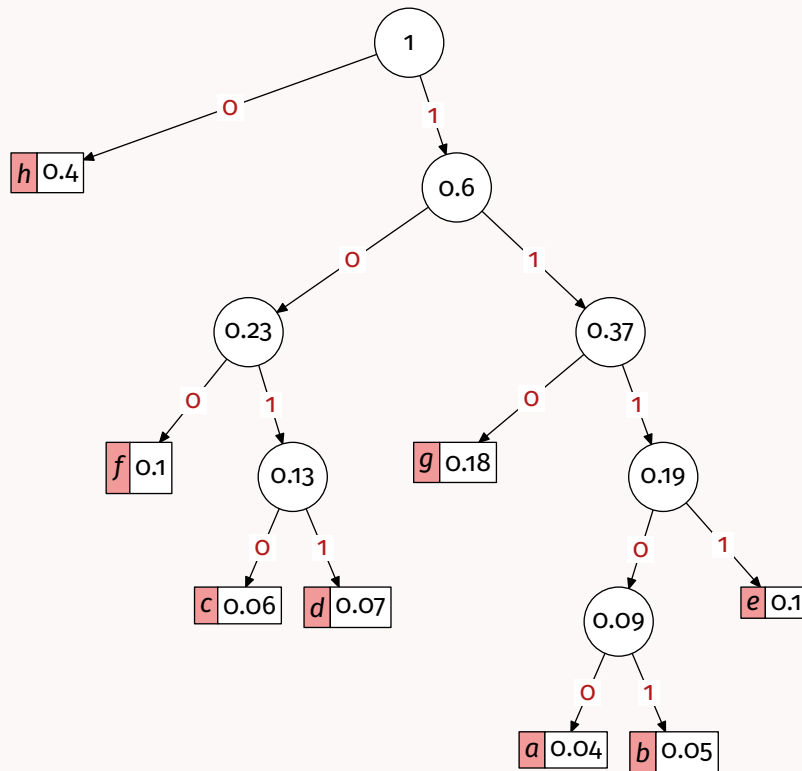
beginfig(0);
string charList[];
numeric frequency[];
charList[1]="a"; frequency[1]:=0.04;
charList[2]="b"; frequency[2]:=0.05;
charList[3]="c"; frequency[3]:=0.06;
charList[4]="d"; frequency[4]:=0.07;
charList[5]="e"; frequency[5]:=0.1;
charList[6]="f"; frequency[6]:=0.1;
charList[7]="g"; frequency[7]:=0.18;
```

```

charList[8]:="h"; frequency[8]:=0.4;

newBinHuffman.myHuff(8)(charList,frequency);
myHuff.c=origin;
drawObj(myHuff);
endfig;

```



### 3 Package Options

You can modify the size of the internal nodes of the tree with the following command:

```
set_node_size(<dim>)
```

**<dim>**: is the diameter of the circle with unity (default: 13pt).

You can change the color for the symbol boxes with the following command:

```
set_leaf_color(<color>)
```

**<color>**: is a METAPOST **color**.

You can hide the bit in the edges of the tree with the following boolean (**true** by default):

`show_bits`

Similarly, you can set the following boolean to `false` to hide the node values:

`show_node_values`

Finally, you can set the following boolean to `false` to hide the leaf values:

`show_leaf_values`

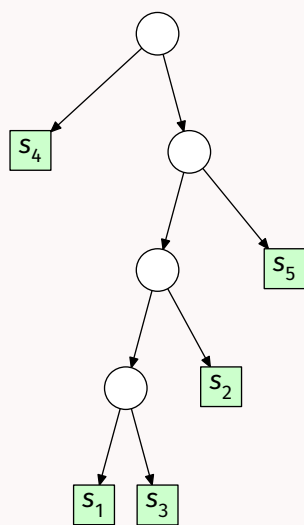
Here an example combining all these commands and variables.

#### Example 2

```
input huffman

beginfig(0);
string charList[];
numeric frequency[];
charList[1]:="s_1"; frequency[1]:=2;
charList[2]:="s_2"; frequency[2]:=4;
charList[3]:="s_3"; frequency[3]:=2;
charList[4]:="s_4"; frequency[4]:=12;
charList[5]:="s_5"; frequency[5]:=8;

set_leaf_color(0.2[white,green]);
set_node_size(8pt);
show_bits:=false;
show_node_values:=false;
show_leaf_values:=false;
newBinHuffman.myHuff(5)(charList,frequency);
myHuff.c=origin;
drawObj(myHuff);
endfig;
```



## 4 METAOBJ Tree Options

Because the Huffman tree is build using METAOBJ tree constructor, the META-  
OBJ tree options are available. All of them are not well suited for this application  
mostly because the Huffman tree is build using elementary trees, to which the  
options we give to the Huffman constructor is passed to all the subtrees.

We give in table 1 the METAOBJ options for the trees that could be used for  
the Huffman constructor.

Option	Type	Default	Description
treemode	string	"D"	direction in which the tree develops; there are four different possible values: "D" (default), "U", "L" and "R"
treeflip	boolean	false	if true, reverses the order of the sub- trees
treenodehsize	numeric	-1pt	if non-negative, all the nodes are as- sumed to have this width
treenodevsize	numeric	-1pt	if non-negative, all the nodes are as- sumed to have this height
dx	numeric	0	horizontal clearance around the tree
dy	numeric	0	vertical clearance around the tree
hsep	numeric	1cm	for a horizontal tree, this is the sepa- ration be- tween the root and the sub- trees
vsep	numeric	1cm	for a vertical tree, this is the separation be- tween the root and the subtrees
hbsep	numeric	1cm	for a vertical tree, this is the horizontal separation between subtrees; the sub- trees are actually put in a HBox and the value of this option is passed to the HBox constructor
vbsep	numeric	1cm	for an horizontal tree, this is the ver- tical separation between subtrees; the subtrees are actually put in a HBox and the value of this option is passed to the HBox constructor
edge	string	"ncline"	name of a connection command (se METAOBJ documentation)
Dalign	string	"top"	vertical alignment of subtrees for trees that go down (the root on the top); the other possible values are "center" and "bot"

Table 1: Table of METAOBJ tree options.

Here is an example of using some of these options.

### Exemple 3

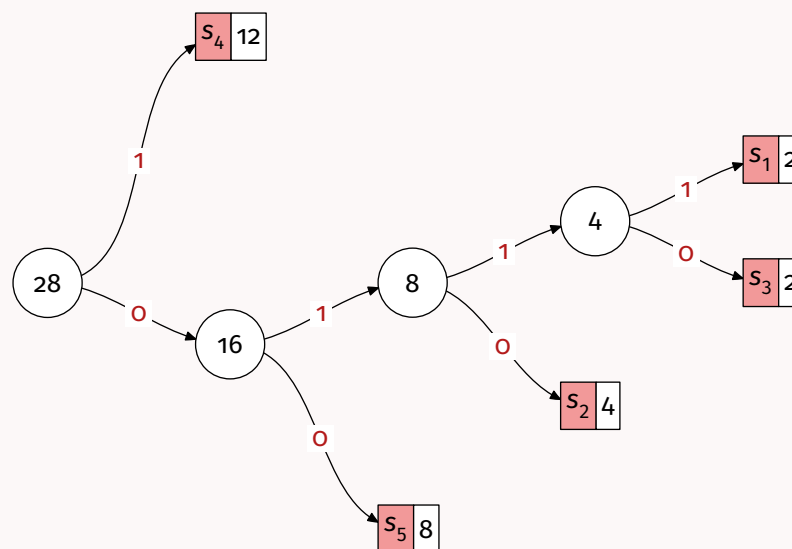
```
input huffman
beginfig(0);
```

```

string charList[];
numeric frequency[];
charList[1]:="s_1"; frequency[1]:=2;
charList[2]:="s_2"; frequency[2]:=4;
charList[3]:="s_3"; frequency[3]:=2;
charList[4]:="s_4"; frequency[4]:=12;
charList[5]:="s_5"; frequency[5]:=8;

newBinHuffman.myHuff(5)(charList,frequency)
"treemode(R)","treeflip(true)","hsep(1.5cm)", "edge(nccurve)" ,
"angleA(0)", "angleB(0)";
myHuff.c=origin;
drawObj(myHuff);
endfig;

```



## 5 Nodes and Subtrees Access

### 5.1 Tree Number

Each time a Huffman tree is built, it is internally counted with a counter. You can get the current Huffman tree number with the following command:

```
get_huffman_tree_nbr()
```

### 5.2 Access to Nodes and Leaves

To access the nodes and the trees, you can use the `treeroot` command from `METAOBJ`.

```
treeroot(Obj(<name>))(<int>,<int>,etc.)
```

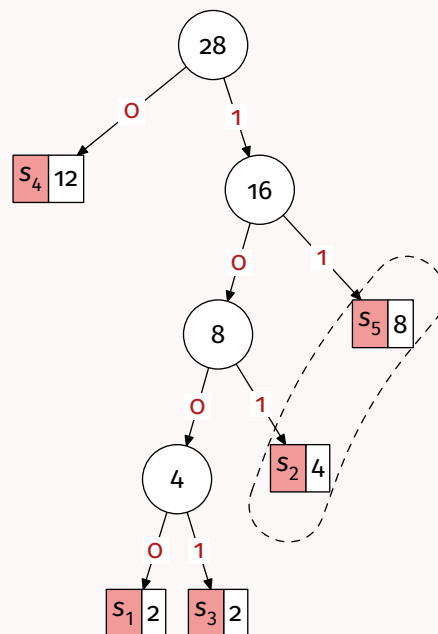
The sequence of  $\langle int \rangle$  gives the choice of branch where the children are numbered from 1 to  $n$ .

The following example shows a use of this mechanism.

#### Example 4

```
input huffman;
beginfig(0);
string charList[];
numeric frequency[];
charList[1]:="s_1"; frequency[1]:=2;
charList[2]:="s_2"; frequency[2]:=4;
charList[3]:="s_3"; frequency[3]:=2;
charList[4]:="s_4"; frequency[4]:=12;
charList[5]:="s_5"; frequency[5]:=8;

newBinHuffman.myHuff(5)(charList,frequency);
myHuff.c=origin;
ncarcbox(treeroot(Obj(myHuff))(2,1,2))(treeroot(Obj(myHuff))
(2,2))
"linestyle(dashed evenly)", "nodesepA(5mm)", "nodesepB(5mm)" ;
drawObj(myHuff);
endfig;
```





## 6 Constructors

### References

- [1] Denis B. Roegel. *The metaobj package. MetaPost package providing high-level objects*. Version 0.93. June 24, 2016. URL: <https://ctan.org/pkg/metaobj>.
- [2] The MetaPost Team and John Hobby. *The metapost package. A development of Metafont for creating graphics*. Aug. 26, 2021. URL: <https://ctan.org/pkg/metapost>.

## Command Index

get\_huffman\_tree\_nbr=, 7

newBinHuffmanTree, 3

set\_leaf\_color, 4

set\_node\_size, 4

show\_bits, 5

show\_leaf\_values, 5

show\_node\_values, 5

treeroot, 7