

Binary Trees (Arbres binaires)

Display and load

ASCII art!

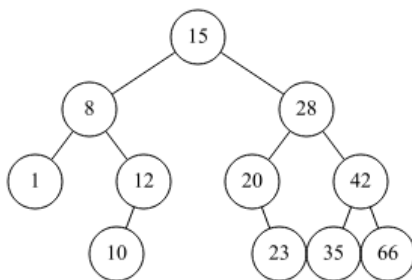
Here are two examples of the same tree displays:

```
1 >>> printTree(B)
2 - 82
3 | - 66
4 | | - 75
5 | | | - 26
6 | | | - 46
7 | | - 59
8 | | | - 45
9 | | - 96
10 | - 13
11 | | - 45
12 | | | - 29
13 | | | - 15
14 | - 47
15 | | - 77
16 | | - 62
```

```
1 >>> prettyPrintTree(B)
2
3      82
4     /  \
5    /    \
6   /      \
7  /        \
8 /          \
9 /            \
10 /            \
11 66          13
12 /  \       /  \
13 /    \   /    \
14 /      \ /      \
15 /        \ 45    47
16 /          \ /    \
17 /            \ 29  15
18 /              \ 77  62
19 26      46    45  96
```

The real pretty display

dot (graphviz)



"Simple" dot:

```
1 graph {
2   15 -- 8
3   15 -- 28
4   8 -- 1
5   8 -- 12
6   28 -- 20
7   28 -- 42
8   12 -- 10
9   20 -- 23
10  42 -- 35
11  42 -- 66
12 }
```

Problem: does not manage identical keys!

Dot with id:

```
1 graph G {
2   N161384992[label="1"];
3   N161385944 -- N161384992;
4   N161385944[label="8"];
5   N161383704 -- N161385944;
6   N161384432[label="10"];
7   N161384768 -- N161384432;
8   N161384768[label="12"];
9   N161385944 -- N161384768;
10  N161383704[label="15"];
11  N161510232[label="20"];
12  N161384376 -- N161510232;
13  N161510288[label="23"];
14  N161510232 -- N161510288;
15  N161384376[label="28"];
16  N161383704 -- N161384376;
17  N161510736[label="35"];
18  N161510064 -- N161510736;
19  N161510064[label="42"];
20  N161384376 -- N161510064;
21  N161513088[label="66"];
22  N161510064 -- N161513088;
23 }
```

Real bonus!

Manage the simple nodes...

Load binary trees

The above tree was loaded from a textfile that contains:

```
1 (15(8(1()())(12(10()())()))(28(20() (23() ())) (42(35() ())(66() ())))))
```

And what about this textfile?

```
1 (A
2   (B
3     (D
4       (H
5         ()
6         ()
7       )
8       (I
9         ()
10        ()
11       )
12     )
13     (E
14       (J
15         ()
16         ()
17       )
18       (K
19         ()
20         ()
21       )
22     )
23   )
24   (C
25     (F
26       (L
27         ()
28         ()
29       )
30       (M
31         ()
32         ()
33       )
34     )
35     (G
36       (N
37         ()
38         ()
39       )
40       (O
41         ()
42         ()
43       )
44     )
45   )
46 )
```

Do not forget to test if the string in the file is "well-formed"!

Both files can be found in <https://algo-td.infoprepa.epita.fr//algo/S2/Python/files/>