1 Binary Tree Diagram

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Egestas purus viverra accumsan in nisl nisi. Arcu cursus vitae congue mauris rhoncus aenean vel elit scelerisque. In egestas erat imperdiet sed euismod nisi porta lorem mollis. Morbi tristique senectus et netus. Mattis pellentesque id nibh tortor id aliquet lectus proin. Sapien faucibus et molestie ac feugiat sed lectus vestibulum. Ullamcorper velit sed ullamcorper morbi tincidunt ornare massa eget. Dictum varius duis at consectetur lorem. Nisi vitae suscipit tellus mauris a diam maecenas sed enim. Velit ut tortor pretium viverra suspendisse potenti nullam. Et molestie ac feugiat sed lectus. Non nisi est sit amet facilisis magna. Dignissim diam quis enim lobortis scelerisque fermentum. Odio ut enim blandit volutpat maecenas volutpat. Ornare lectus sit amet est placerat in egestas erat. Nisi vitae suscipit tellus mauris a diam maecenas sed. Placerat duis ultricies lacus sed turpis tincidunt id aliquet.

```
# https://github.com/BaseMax/BinaryTreeDiagram
import itertools, graphviz as gvz
from math import log2, floor
matrix = {
    "xyz": (0, 1, 2, 3, 4, 5, 6, 7),
    "xzy": (0, 1, 4, 5, 2, 3, 6, 7),
    "yxz": (0, 1, 4, 5, 2, 3, 6, 7),
    "yzx": (0, 4, 1, 5, 2, 6, 3, 7),
    "zxy": (0, 2, 4, 6, 1, 3, 5, 7),
    "zyx": (0, 4, 2, 6, 1, 5, 3, 7),
}

combf = list(map(lambda x: int(x),input('Enter all of minterms in one line with space:').split()))
combf.sort()


def merge(lst):
    res = []
    for i in range(0, len(lst) - 1, 2):
        res.append((lst[i], lst[i + 1]))
    return res


def make_form(combf, fulltree):
    res = []
    for i in range(len(matrix[form])):
        if matrix[form][i] in combf:
            res.append(matrix[form][i])
        else:
            res.append(None)
    while (len(res) > 1):
        res = merge(res)
    return res[0]


def find_best_poly(combf, fulltree, w=0):
```

```
37      new_combf = []
38      for element in combf:
39          if element in fulltree:
40              new_combf.append(element)
41              w += 1
42      if len(new_combf) < 2:
43          return w
44      return find_best_poly(list(itertools.combinations(new_combf, 2)
            ),merge(fulltree), w)
45
46
47  def draw(tree, g, form, h=1):
48      if (hasattr(tree, "__iter__")):
49
50          l = draw(tree[0], g, form, 2 * h)
51          r = draw(tree[1], g, form, 2 * h + 1)
52
53          if l == r and r != (1, 1):
54
55              g.node(f'{h}',f'{l[0]}',style="invis" if l==(0,0) else
                    None)
56
57              g.node(f'{2*h}',f'{l[0]}',style="invis")
58              g.edge(f'{h}',f'{2*h}',style="invis")
59
60              g.node(f'{2*h+1}',f'{r[0]}',style="invis")
61              g.edge(f'{h}',f'{2*h+1}',style = "invis")
62
63              return l
64
65          g.node(f'{h}', f'{form[floor(log2(h))]}')
66
67          g.node(f'{2*h}',f'{form[floor(log2(2*h))]}' if l[1] else f
                '{l[0]}' ,style=None if l[0] else "invis")
68          g.edge(f'{h}', f'{2*h}',style="dashed" if l[0] else "invis"
                )
69
70          g.node(f'{2*h+1}',f'{form[floor(log2(2*h+1))]}' if r[1]
                else f'{r[0]}',style=None if r[0] else "invis")
71          g.edge(f'{h}', f'{2*h+1}',style=None if r[0] else "invis")
72
73          return (1, 1)
74
75      return (1, 0) if type(tree) == int else (0, 0)
76
77
78
79  if __name__ == "__main__":
80      weight = {
81          fulltree[0]: find_best_poly(list(itertools.combinations(
                combf, 2)),merge(fulltree[1]))
82          for fulltree in matrix.items()
83      }
84      form = list(weight.keys())[list(weight.values()).index(max(
            weight.values()))]
85      print(form)
86      g = gvz.Graph(format="png",filename="btree.gv")
```

```
87        draw(make_form(combf, matrix[form]), g, form)
88        print(g.source)
89        g.view()
```

References

[1] Source code of program: Binary Tree Diagram Drawing, Max Base
`https://github.com/BaseMax/BinaryTreeDiagram`