6lo WG
@IETF 113

Demonstration of NSA Allocation Function

draft-li-6lo-native-short-address-02

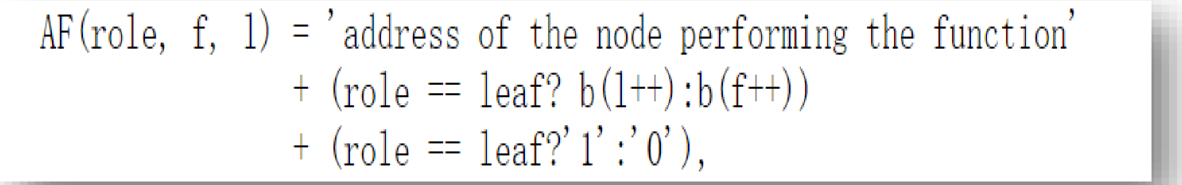
Guangpeng Li

March, 2022

Specifications on NSA Allocation Functionn

- The allocation function for NSA can be different case by case, but all nodes under the same root MUST use the same one. See section 4

- The typical example algorithm given in the draft is as follows:

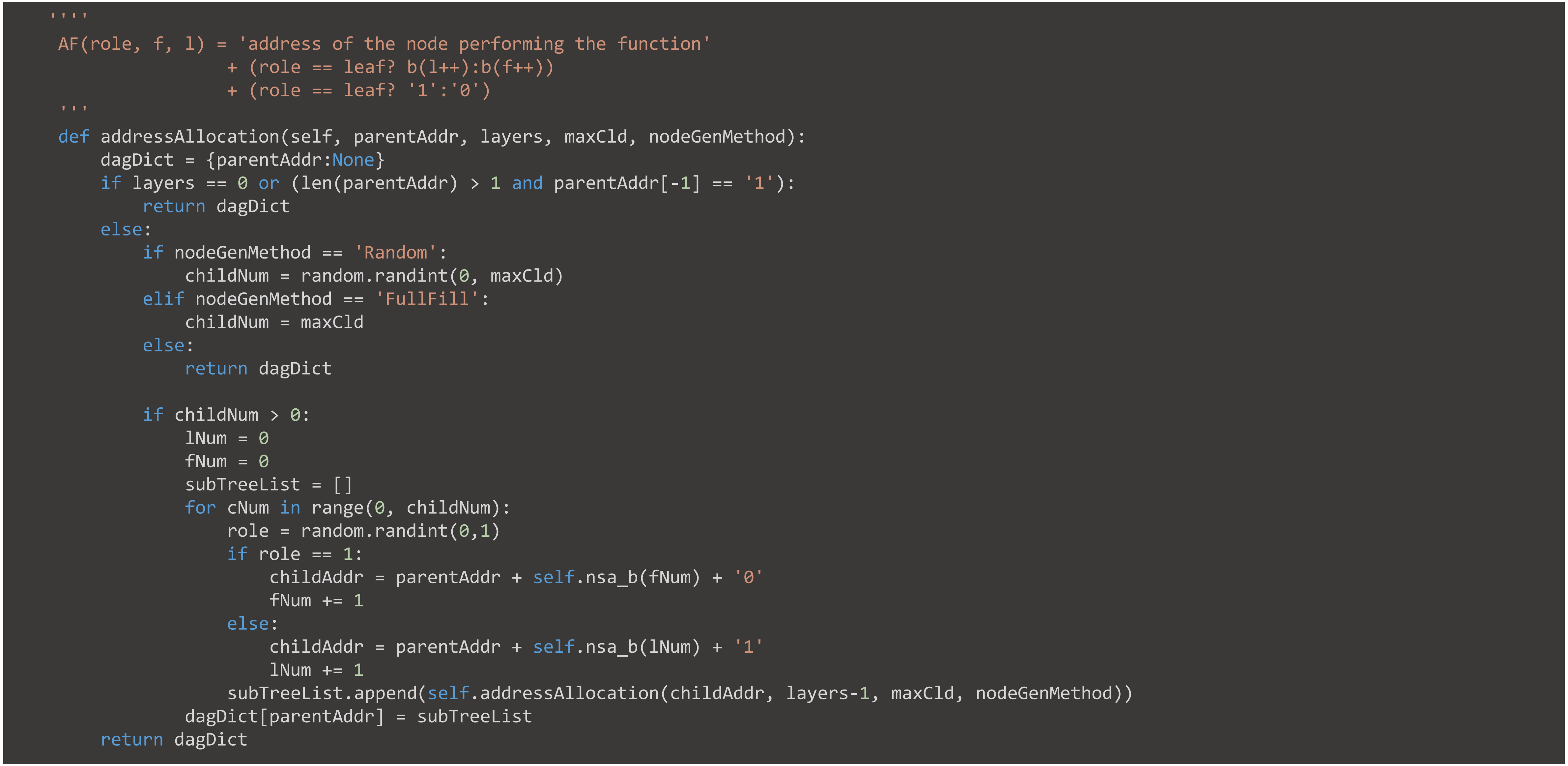


```
AF(role, f, l) = ’address of the node performing the function’
               + (role == leaf? b(l++):b(f++))
               + (role == leaf?’1’:’0’),
```

- In section 4, the formula to calculate max length of addresses is given:

  Max_Length = length(Parent address) + length(b(max(f,l))) + 1

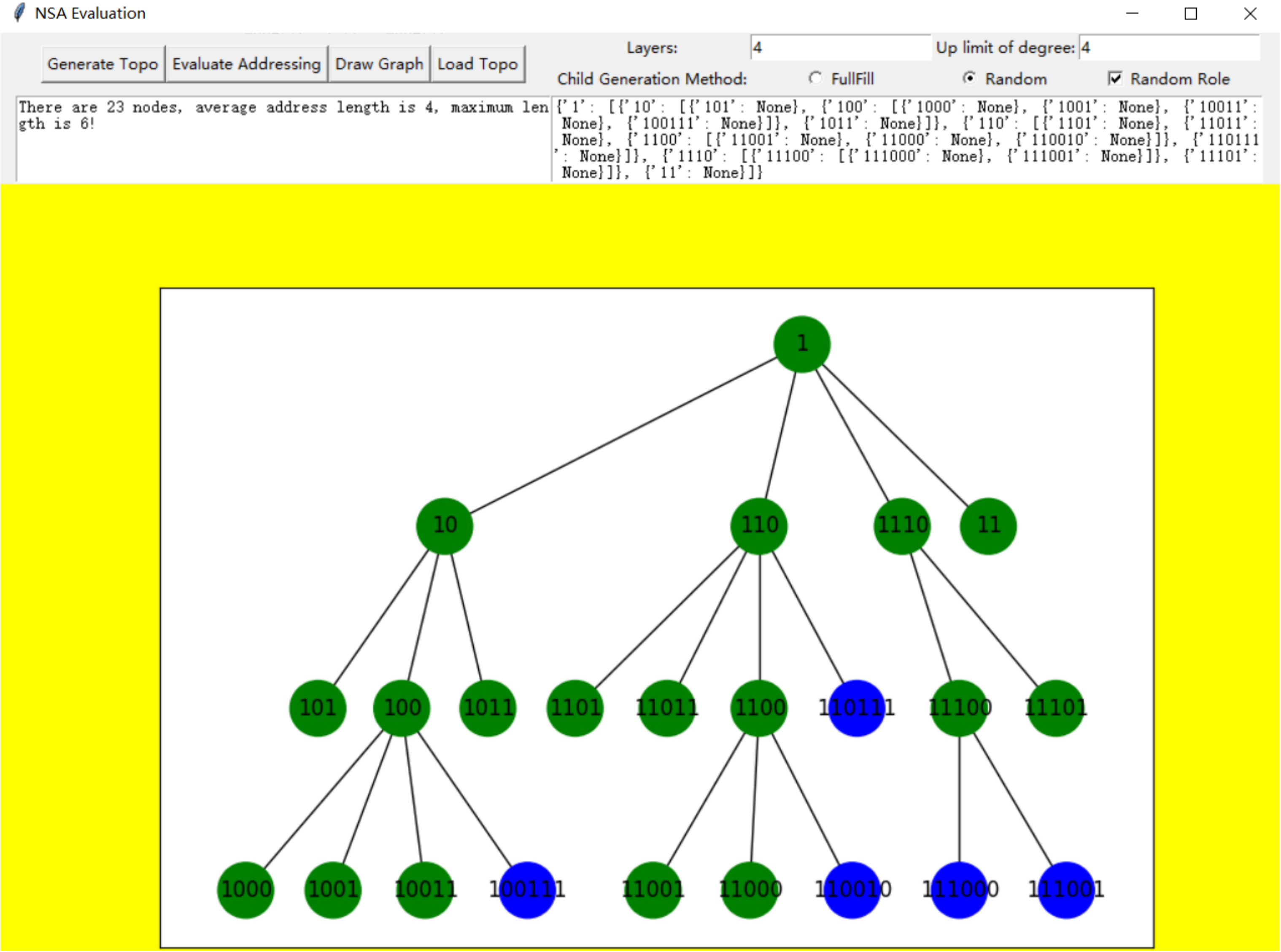
The Code of AF example in the draft



```
''''
 AF(role, f, l) = 'address of the node performing the function'
                 + (role == leaf? b(l++):b(f++))
                 + (role == leaf? '1':'0')
 '''
 def addressAllocation(self, parentAddr, layers, maxCld, nodeGenMethod):
     dagDict = {parentAddr:None}
     if layers == 0 or (len(parentAddr) > 1 and parentAddr[-1] == '1'):
         return dagDict
     else:
         if nodeGenMethod == 'Random':
             childNum = random.randint(0, maxCld)
         elif nodeGenMethod == 'FullFill':
             childNum = maxCld
         else:
             return dagDict

         if childNum > 0:
             lNum = 0
             fNum = 0
             subTreeList = []
             for cNum in range(0, childNum):
                 role = random.randint(0,1)
                 if role == 1:
                     childAddr = parentAddr + self.nsa_b(fNum) + '0'
                     fNum += 1
                 else:
                     childAddr = parentAddr + self.nsa_b(lNum) + '1'
                     lNum += 1
                 subTreeList.append(self.addressAllocation(childAddr, layers-1, maxCld, nodeGenMethod))
             dagDict[parentAddr] = subTreeList
     return dagDict
```

Case 1: Generate a topology and Assign Address



Layers: 4

Up limit of degree: 4

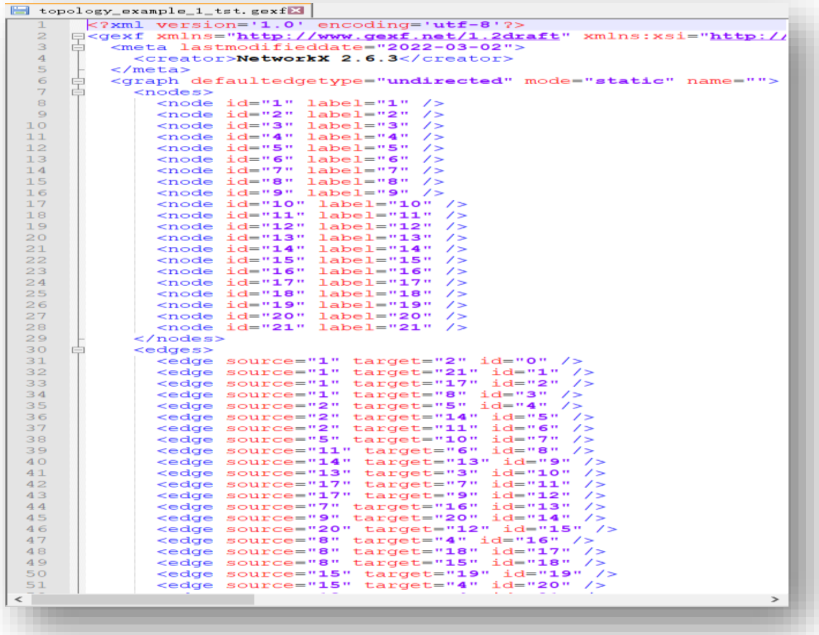
Child Generation Method: ○ FullFill ● Random ☑ Random Role

```
There are 23 nodes, average address length is 4, maximum len
gth is 6!
```

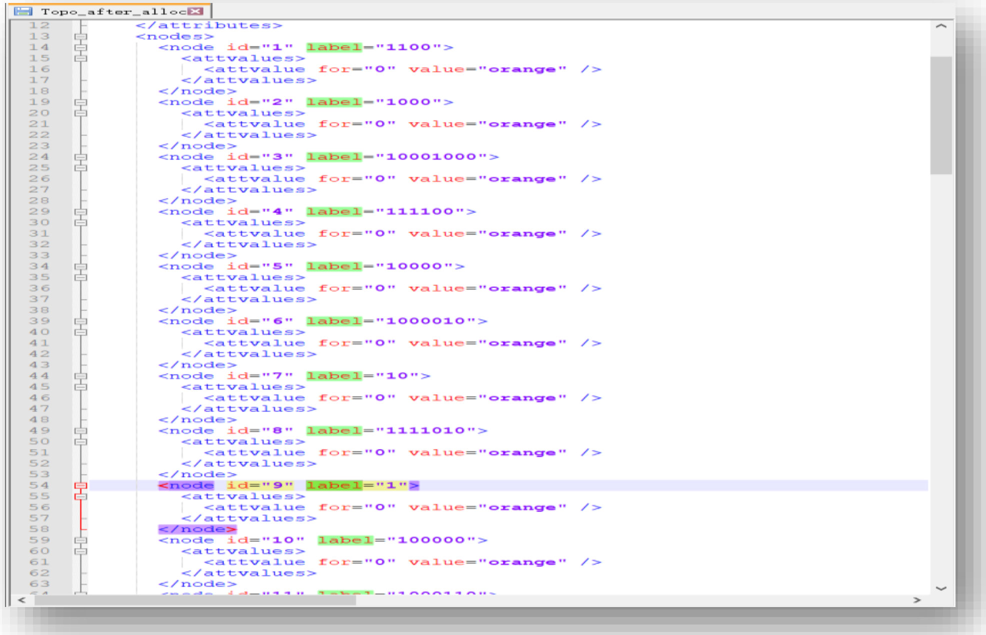
```
{’1’: [{’10’: [{’101’: None}, {’100’: [{’1000’: None}, {’1001’: None}, {’10011’:
 None}, {’100111’: None}]}, {’1011’: None}]}, {’110’: [{’1101’: None}, {’11011’:
 None}, {’1100’: [{’11001’: None}, {’11000’: None}, {’110010’: None}]}, {’110111
’: None}]}, {’1110’: [{’11100’: [{’111000’: None}, {’111001’: None}]}, {’11101’:
 None}]}, {’11’: None}]}
```

- Input total layers in the tree and maximum number of children of each node. In left case, the parameters are 4 and 4.
- Quantity of child is determined randomly here
- After evaluation, there are totally 23 nodes
- Maximum address length is 6, owned by blue nodes.
- Average length of addresses is 4 in this case

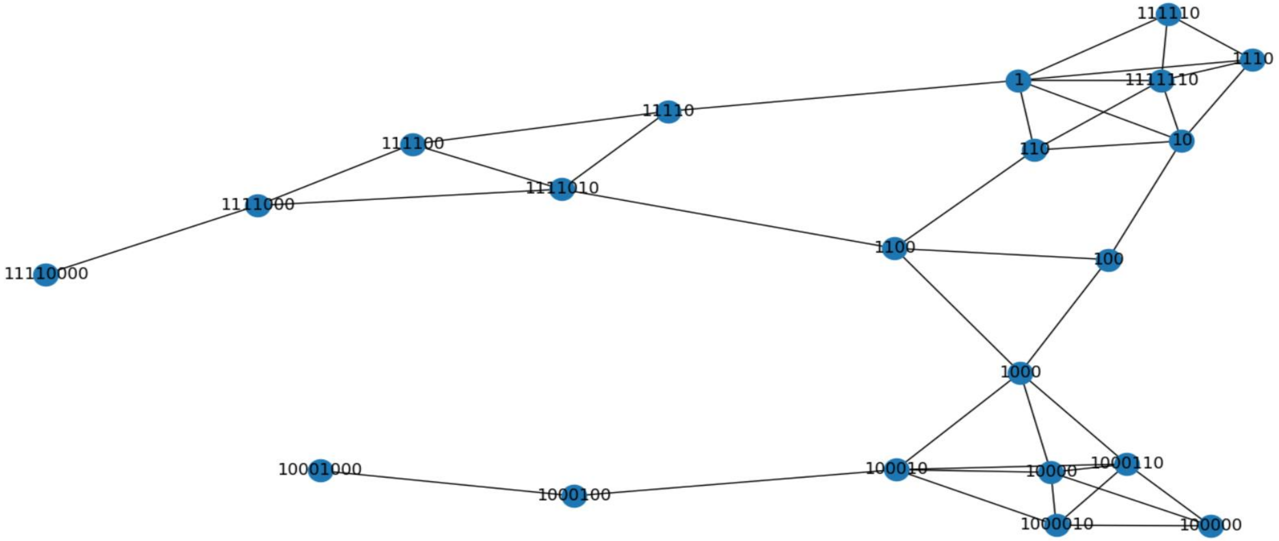
Case 2: Read arbitrary graph and allocate NSA addresses



```
<?xml version='1.0' encoding='utf-8'?>
<gexf xmlns="http://www.gexf.net/1.2draft" xmlns:xsi="http://
  <meta lastmodifieddate="2022-03-02">
    <creator>NetworkX 2.6.3</creator>
  </meta>
  <graph defaultedgetype="undirected" mode="static" name="">
    <nodes>
      <node id="1" label="1" />
      <node id="2" label="2" />
      <node id="3" label="3" />
      <node id="4" label="4" />
      <node id="5" label="5" />
      <node id="6" label="6" />
      <node id="7" label="7" />
      <node id="8" label="8" />
      <node id="9" label="9" />
      <node id="10" label="10" />
      <node id="11" label="11" />
      <node id="12" label="12" />
      <node id="13" label="13" />
      <node id="14" label="14" />
      <node id="15" label="15" />
      <node id="16" label="16" />
      <node id="17" label="17" />
      <node id="18" label="18" />
      <node id="19" label="19" />
      <node id="20" label="20" />
      <node id="21" label="21" />
    </nodes>
    <edges>
      <edge source="1" target="2" id="0" />
      <edge source="1" target="21" id="1" />
      <edge source="1" target="17" id="2" />
      <edge source="1" target="8" id="3" />
      <edge source="2" target="5" id="4" />
      <edge source="2" target="14" id="5" />
      <edge source="2" target="11" id="6" />
      <edge source="5" target="10" id="7" />
      <edge source="11" target="6" id="8" />
      <edge source="14" target="13" id="9" />
      <edge source="13" target="3" id="10" />
      <edge source="17" target="7" id="11" />
      <edge source="17" target="9" id="12" />
      <edge source="7" target="16" id="13" />
      <edge source="9" target="20" id="14" />
      <edge source="20" target="12" id="15" />
      <edge source="8" target="4" id="16" />
      <edge source="8" target="18" id="17" />
      <edge source="8" target="15" id="18" />
      <edge source="15" target="19" id="19" />
      <edge source="15" target="4" id="20" />
```



```
      </attributes>
      <nodes>
        <node id="1" label="1100">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="2" label="1000">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="3" label="10001000">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="4" label="111100">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="5" label="10000">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="6" label="1000010">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="7" label="10">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="8" label="1111010">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="9" label="1">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
        <node id="10" label="100000">
          <attvalues>
            <attvalue for="0" value="orange" />
          </attvalues>
        </node>
```



Live Demo

THANKS!