◁ ◁ ◁ Rapport ▷ ▷ ▷

◇ ◇ ◇

# Projet S8 - Le PQ Tree

⋆ ⋆ ⋆

**Tuteur :PREA Pascal**

LI Guodong.

Élève-ingénieur à l'École Centrale de Marseille

Mai 2017

# Introduction

A PQ tree is a tree-based data structure that represents a family of permutations on a set of elements, discovered and named by Kellogg S. Booth and George S. Lueker in 1976. It is a rooted, labeled tree, in which each element is represented by one of the leaf nodes, and each non-leaf node is labelled P or Q. A P node has at least two children, and a Q node has at least three children.(Wikipedia)

There are three types of nodes for the PQ tree. The P node is described by the shape of circle. The Q node is described by the shape of rectangle. The leaf is described by the number.

One PQ tree can be described as many types of form. Because we can reorder the sub nodes of some P nodes arbitrarily and reverse the order of the sub nodes of some Q nodes. The frontier of a PQ tree is the set of leaves, read in left-to-right order. With general PQ tree, we can restrict the PQ tree for some specific problems in real life. For example, we can use it to the problems : determining whether a graph is planar, DNA Sequence Reconstruction and Tagging the Clones.

The aim of my project is drawing the PQ Tree automatically. That is to say, when you input the information of PQ tree, I can use my programme to draw the PQ Tree exactly as you want.

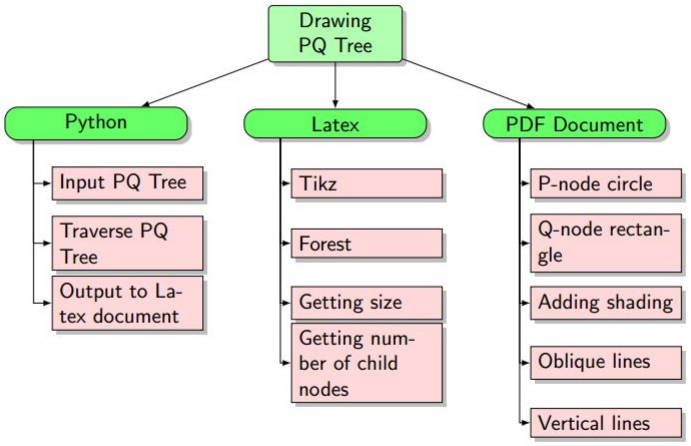


FIGURE 1 – Principal : explicate the process of the project

The figure above explicate the process of my project. After, I will describe the details of my project. The basic function is to generate the graph of PQ Tree automatically. After implementing this function I have tried to make the PQ Tree generated more beautiful. That is to say, I want to not only realize the basic function but also make it more reasonable.

I will describe the structure evolution of my PQ Tree. The change includes three structures of PQ Tree along with the features.

# 1 Basic questions to solve

## 1.1 How to send the information of PQ Tree by the way : Python–>Latex document–>PDF document

First, for the programme python, I used multi-list to input the information of PQ Tree. For example, ['Q-node', 8, 5, 4, 7, 1, ['P-node', 0, ['Q-node', 1, 3], 9, 2, 6], 1, 2]. Then, for sending the information to latex, the following programme python was used to open a document latex to operate.

```
if FILE_NAME is not None and len(FILE_NAME) > 0:
    THE_FILE = open(FILE_NAME + ".tex", "w")
else:
    THE_FILE = None
```

FIGURE 2 – the function to open a latex document by python

Finally, I could get the expected information in the document latex and generate the PDF document from latex document directly. That is to say, I changed nothing after generating the latex document.All the basic settings and methods were finished by the Python programme

## 1.2 How to traverse the PQ Tree

To traverse the PQ Tree in my python program is to traverse the multi-list. The recursive method is used to traverse the PQ Tree perfectly.

There are 3 important elements for a PQ Tree. When I have a element with type 'string', it is either P-node or Q-node. I can output the corresponding phrase to latex document.At the same time, it is the symbol of a new branch to generate, for which the function should be executed again. When I have a element with type 'integer', the corresponding phrase with correct form will also be printed in the latex document to represent the leaves of PQ Tree. The function 'gutemberg' is written to print the corresponding phrase in the latex document.The lines of figures will be talked about in the second part.

```
def draw_pi_qew_tree(pi_qew_tree):
    for i in pi_qew_tree:
        # print(str(type(i)))
        if type(i) is str:
            if i == 'P_node':
                gutemberg("[,ellipse=" + str(traversal_by_int(new_list2[a])) + ",*|")
            if i == 'Q_node':
                gutemberg("[,rect=" + str(traversal_by_int(new_list2[a])) + ",*|")
        elif type(i) is int:
            gutemberg("[" + str(i) + "]")
        elif type(i) is list:
            a += 1  # global variable
            draw_pi_qew_tree(i)
            gutemberg("]")
```

FIGURE 3 – the function to traverse the PQ Tree

# 2 Structure evolution process of the PQ Tree

## 2.1 Version 1

The latex package 'forest' and 'tikz' were used to represent the PQ Tree. P-node is reprensented by the shape of circle and Q-node is reprensented by the shape of square. All the connected lines are oblique lines.It is the start version of my project.

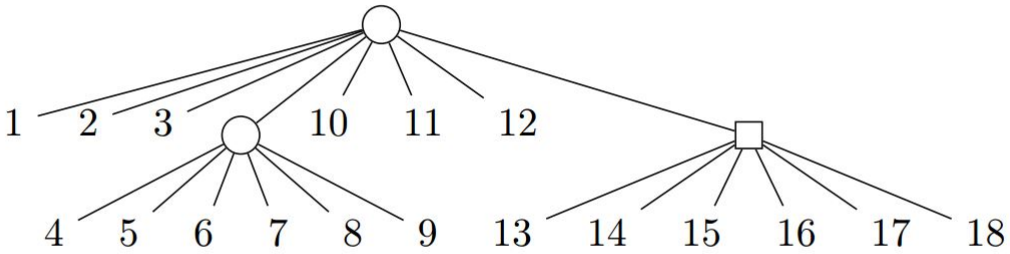


FIGURE 4 – PQ Tree version1

The advantage of this version is that all the nodes will be generated automatically and quickly. There will be no problem of overlapping. The disadvantage is that if the PQ Tree is too complicated, the connected lines maybe too crowded.

## 2.2 Version 2

The second version is the interim version, of which vertical lines are always with Q-node(rectangle) and oblique lines are always with P-node(circle).The shape of Q-node was changed to rectangle.

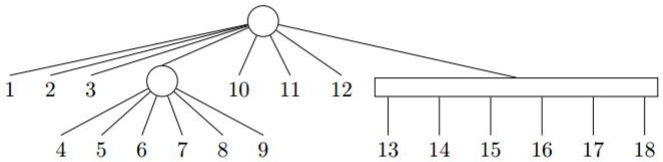


FIGURE 5 – PQ Tree version2

The advantage of this version is that the difference between the P-node and Q-node is obvious so that we can tell the difference easily. The disadvantage is that the size of P-node and Q-node is very

different, so maybe it looks not very beautiful.

### 2.3 Version 3

The third version is the final version, of which all the lines are vertical and all the numbers are on the same level at the bottom which looks more balanced and beautiful. The advantage of this version is that

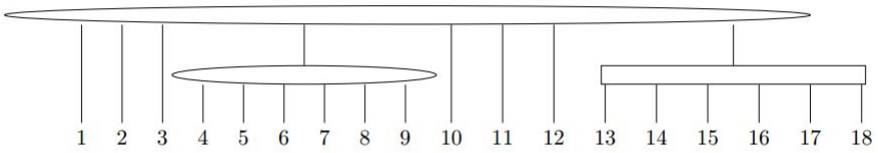


FIGURE 6 – PQ Tree version3

all the nodes will be generated automatically with which all the lines are vertical and all the numbers are on the same level at the bottom. The effect will be beautiful. The disadvantage is that there maybe exist the problem of overlapping.It is not easy to tell difference between P-node and Q-node.

So, I have added the function to make the difference between the P-node and Q-node more obvious. The function is adding the shading effect as you like to either P-node or Q-node, which will be shaded by the oblique lines.

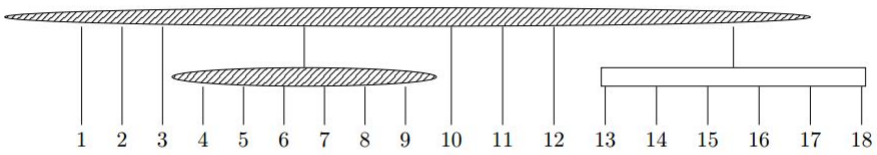


FIGURE 7 – PQ Tree version3, shading effect with P-node

## Conclusion

My project has achieved the aim to draw the PQ Tree you want immediately. The advantage of Python is to calculate and the advantage of Latex is to draw beautifully. I have used the advantages of them and combine them together to make the process quickly, correctly and beautifully.