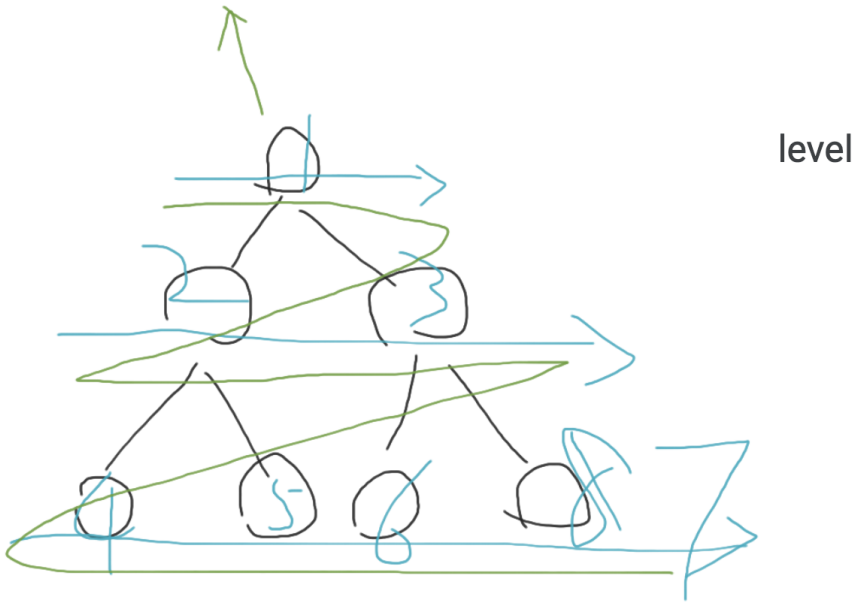
Phill-DS-0124, 0201

BFS (Breath-First Search)

- traverse 方式
- 本質上是Queue的問題
  - TreeNode → Node
  - QueueNode → 根據綠線來連結
  - Queue 框架




```
#include <iostream>
using namespace std;

struct TreeNode{
  int data;
  TreeNode *llink;
```

```
  TreeNode *rlink;

  Tree(int val): data(val), llink(nullptr), rlink(nullptr) {}
};

struct QueueNode{ //不自帶資料
  TreeNode* treeNode;
  QueueNode* next;

  QueueNode(TreeNode* node): treeNode(node), next(nullptr) {}
};

class Queue{
  private:
    QueueNode* front;
    QueueNode* rear;

  public:
    Queue(): front(nullptr), rear(nullptr) {}

    ~Queue(){
      while(front!= nullptr){
        QueueNode* temp= front; //ready to release
        front = front ->next; // 指向下一個
        delete temp;
      }
    }

    void enqueue(TreeNode* node){
      QueueNode* newNode = new QueueNode(node);
      if(rear == nullptr){
        front = rear = newNode;
        return;
      }
      rear->next = newNode;
      rear = newNode;
```

```
    }

    TreeNode* dequeue(){
      if(front==nullptr)
        return nullptr;
      QueueNode* temp = front;
      front = front -> next;
      if(front==nullptr)
        rear=nullptr;
      TreeNode* result= temp->treeNode;
      delete temp;
      return result;
    }

        //Queue isEmpty
        bool isEmpty(){
            return front == nullptr;
        }
};

class BinaryTree{
  private:
    TreeNode* root;

  public:
    BinaryTree(): root(nullptr) {}

    void buildBFS(){
      if(root==nullptr)
        return;
      Queue q;
      q.enqueue(root); //root -> queue
      while(){ //橫跨黑色樹 綠色結構
        TreeNode* current = q.dequeue(); //黑色節點
        if(current->llink !=nullptr)
          q.enqueue(current->llink);
```

```
        if(current->rlink !=nullptr)
          q.enqueue(current->rlink);
      }
    }

        TreeNode* BFSearch(int value){
          if(root==nullptr)
            return;
          Queue q;
          q.enqueue(root); //root -> queue, 初始迭代的第一個數值 ro
          while(!q.isEmpty()){ //橫跨黑色樹 綠色結構 -> 迭代的單元動
            TreeNode* current = q.dequeue(); //黑色節點
                    if(current->data == value) return current;
            if(current->llink !=nullptr)
              q.enqueue(current->llink);
            if(current->rlink !=nullptr)
              q.enqueue(current->rlink);
          }
        }
};


int main()
{

    return 0;
}
```