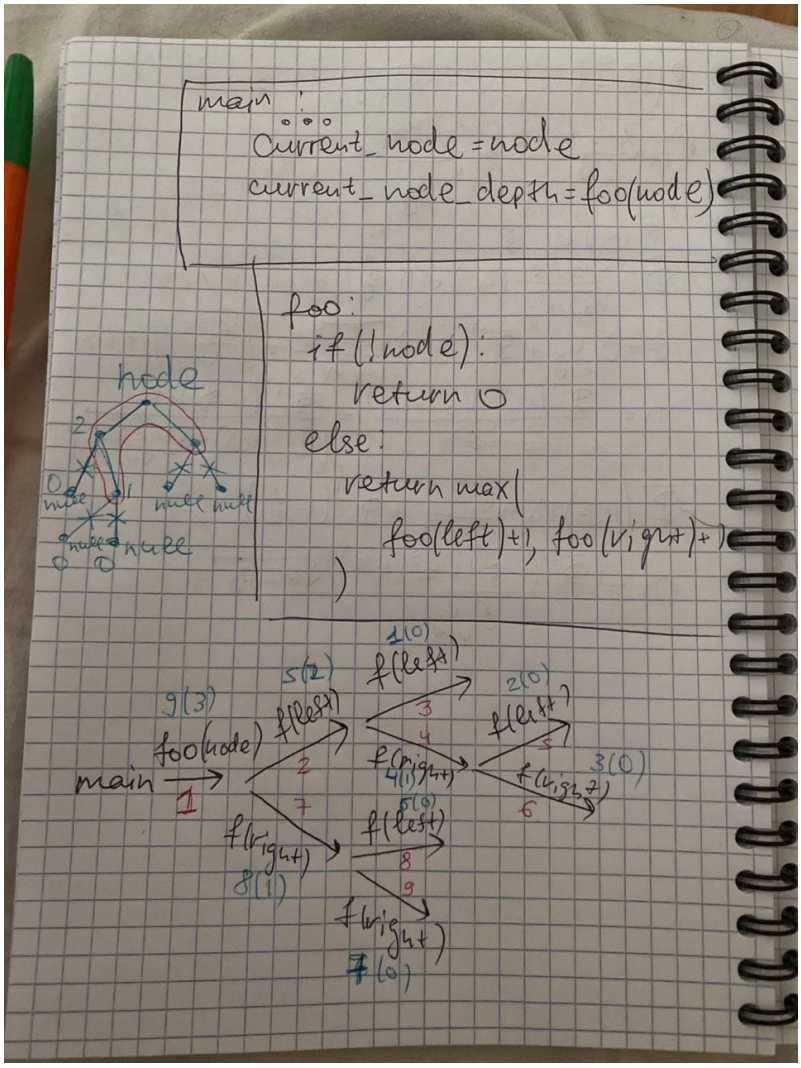
btree_level_count

Пример дерева, для которого мы вызываем функцию. Все зеленые точки слева - это то, где ты попробуешь вызвать функцию



Сверху черным две функции

```
int
btree_level_count(t_
btree *root)
{
  int  size_l;
  int  size_r;

  if (!root)
    return (0);
  size_l =
btree_level_count(ro
ot->left);
  size_r =
btree_level_count(ro
ot->right);
  if (size_l > size_r)
    return (size_l + 1);
  else
    return (size_r + 1);
}
```

Мы сейчас в корне этого дерева
Вызываем функцию
Функция отрабатывает строка-за строкой
Объявил переменные
Проверил, что ты стоишь в реальном узле (не null). (Почему ты вообще можешь стоять в null? - Потому что у листьев левый и правый ребенок null).
В нашем случае это корень дерева, всё, что выше, нас не волнует
Ты сейчас не в null
Значит ты в реальном узле
Значит у тебя есть left и right
**Вызываешь функцию для left
Внимание! Вызывается новая функция от другого аргумента (в данном случае для left)
Новая функция объявляет новые переменные
Новая функция проверяет, что ее аргумент не null
Новая функция убеждается, что она в реальном узле
И если она в реальном узле, она тоже вызывает left
Допустим, что она не в реальном узле оказалась, то есть попала в null
Тогда она не вызывала left, а вернула 0
(См **): И вот пока она не вернула что-то, здесь не происходит перехода к следующей строке
Очередность вызовов функций ты можешь посмотреть на рисунке снизу (красным цветом)
Зеленым цветом - очередность выходов из вызываемых функций и возвращаемые значения

i