Задание 1

> Даны m элементов в отсортированном. Удалите их из 2-3-дерева за $O(m \log n)$ work и $O(\log m(\log n + \log m))$ span. (Псевдокод не нужен, опишите просто идею)

Пусть M - изначальный массив длины m. T 2-3-дерево.

1. Пометка вершин.
   - Для полуинтервала $[l, r)$ индексов M берем центральный c $= \frac{l+r}{2}$ элемент M и ищем его в T, в процессе поиска помечаем все посещенные вершины, которые содержат в ключе элемент `M[c]`, в лучшем случае помечен будет один лист,в худшем весь путь из $\log n$ вершин. Далее рекурсивно запускаем поиск на $[l, m)$ и $[m, r)$ в `fork_2_join`.
   - Запустив этот алгоритм на $[0, m]$ получим все размеченные для удаления вершины на

$$\text{work} = O(m * \log(n)), \text{span} = O(m * \log(n))$$

2. Удаление вершин.
   - Если вершина помечена то удаляем ее из графа
   - Если нет и ее родитель был удален, то добавляем вершину в массив поддеревьев
   - Запускаем рекурсивно `fork_2_join` на сыновьях, возвращаем объеденный массив вершин поддеревьев

$$\text{work} = O(n), \text{span} = O(\log(n))$$

3. Слияние поддеревьев

Задание 2

> Постройте 2-3 дерево по отсортированному массиву из n элементов за $O(n)$ work и $O(\log n)$ span.

```go
func Build23Tree(data []int) *Node {
  return buildNode(data, 0, len(data))
}

func buildNode(data []int, l, r int) *Node {
  n := r - l
  if n == 1 {
    return &Node{Key: data[l]}
  }

   var children []*Node

   if n%3 == 1 || n == 2 {
     var left, right *Node
     m := (l + r) / 2
     Fork2Join(
       func() { left = buildNode(data, l, m) },
       func() { right = buildNode(data, l, m) },
     )
     children = []*Node{left, right}
   } else {
     var left, middle, right *Node
     m1 := l + n/3
```

```
24      m2 := l + 2*n/3
25      Fork2Join(
26        func() { left = buildNode(data, l, m1) },
27        func() { middle = buildNode(data, m1, m2) },
28        func() { right = buildNode(data, m2, r) },
29      )
30      children = []*Node{left, middle, right}
31    }
32
33    return &Node{Children: children}
34  }
```