

## 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. Параллельно разбиваем дерево на  $m + 1$  поддеревьев  $(T_1, \dots, T_{m+1})$ , которые содержат  $M$  элементы на границах. Разбиваем  $M$  пополам, ищем центральный элемент в дереве  $O(\log n)$ , помечаем путь до него. Вызываем рекуррентно алгоритм с помощью Fork2Join на левой и правой половинке  $M$ , глубина рекурсии  $O(\log m)$

$$\text{work} = m \cdot \log n, \text{span} = \log m \log n$$

2. Параллельно выкидываем помеченные элементы из поддеревьев  $T_i$

$$\text{work} = m, \text{span} = \log m$$

3. Мерджим поддеревья обратно. Параллельно мерджим пары  $(T_1, T_2)$ ,  $(T_3, T_4)$ , etc. потом  $(T_{12}, T_{34})$  и тд. Количество мерджий:  $O(\log m)$ , стоимость мерджа  $O(\log m + \log n)$  (Деревья будут отличаться по краям)

$$\text{work} = m \cdot \log n, \text{span} = \log m(\log m + \log n)$$

## Задание 2

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

```

1 func Build23Tree(data []int) *Node {
2     leaves := make([]*Node, len(data))
3     ParallelFor(
4         0, len(data),
5         func(i int) { leaves[i] = &Node{Key: data[i]} },
6     )
7
8     level := buildLevel(leaves)
9
10    return level[0]
11 }
12
13 func buildLevel(nodes []*Node) []*Node {
14     if len(nodes) == 1 { // корень
15         return nodes
16     }
17
18     n := len(nodes)
19     levelSize := (n + 2) / 3
20     level := make([]*Node, levelSize)
21
22     ParallelFor(0, levelSize, func(i int) {
23         start := i * 3
24         end := min(start+3, n)
25         level[i] = &Node{Children: nodes[start:end]}
26     })
27

```

```
28     return buildLevel(level)
29 }
```

### Задание 3\*

Придумайте такие асимптотики для алгоритмов, чтобы имело смысл использовать 3 уровня accelerating cascades.

1. R, который решает задачу за  $O(RW(n))$  work и  $O(RS(n))$  span.
2. Q, который уменьшает задачу в  $QX = O(1)$  раз, за  $O(QW(n))$  work и  $O(QS(n))$  span.
3. R, который решает задачу за  $O(RW(n))$  work и  $O(RS(n))$  span.

Я пас