Билет 22. Introsort (Интроспективная сортировка)

Билет 22. Introsort

Основная идея: гибридный алгоритм, который начинает как QuickSort, но следит за собой и переключается на HeapSort при проблемах

Что такое Introsort?

Introsort (от "introspective sort"— интроспективная сортировка) — гибридный алгоритм, который сочетает сильные стороны трех алгоритмов:

- **QuickSort** - скорость в среднем случае
- **HeapSort** - гарантии в худшем случае
- **Insertion Sort** - эффективность на маленьких массивах

Компоненты Introsort

1. Быстрая сортировка (QuickSort)

- **Плюсы:** O(n log n) в среднем, очень быстрая на практике
- **Минусы:** $O(n^2)$ в худшем случае
- **Используем:** Как основной алгоритм

2. Пирамидальная сортировка (HeapSort)

- **Плюсы:** Гарантированное O(n log n) даже в худшем случае
- **Минусы:** Медленнее QuickSort в среднем случае
- **Используем:** Как "страховку"от худшего случая

3. Сортировка вставками (Insertion Sort)

- **Плюсы:** O(n) на почти отсортированных данных, быстрая на маленьких массивах
- **Минусы:** $O(n^2)$ в общем случае
- **Используем:** Для маленьких подмассивов

Алгоритм Introsort

Шаг 1: **Инициализация:** Вычисляем максимальную глубину рекурсии: $2 \cdot \lceil \log_2 n \rceil$

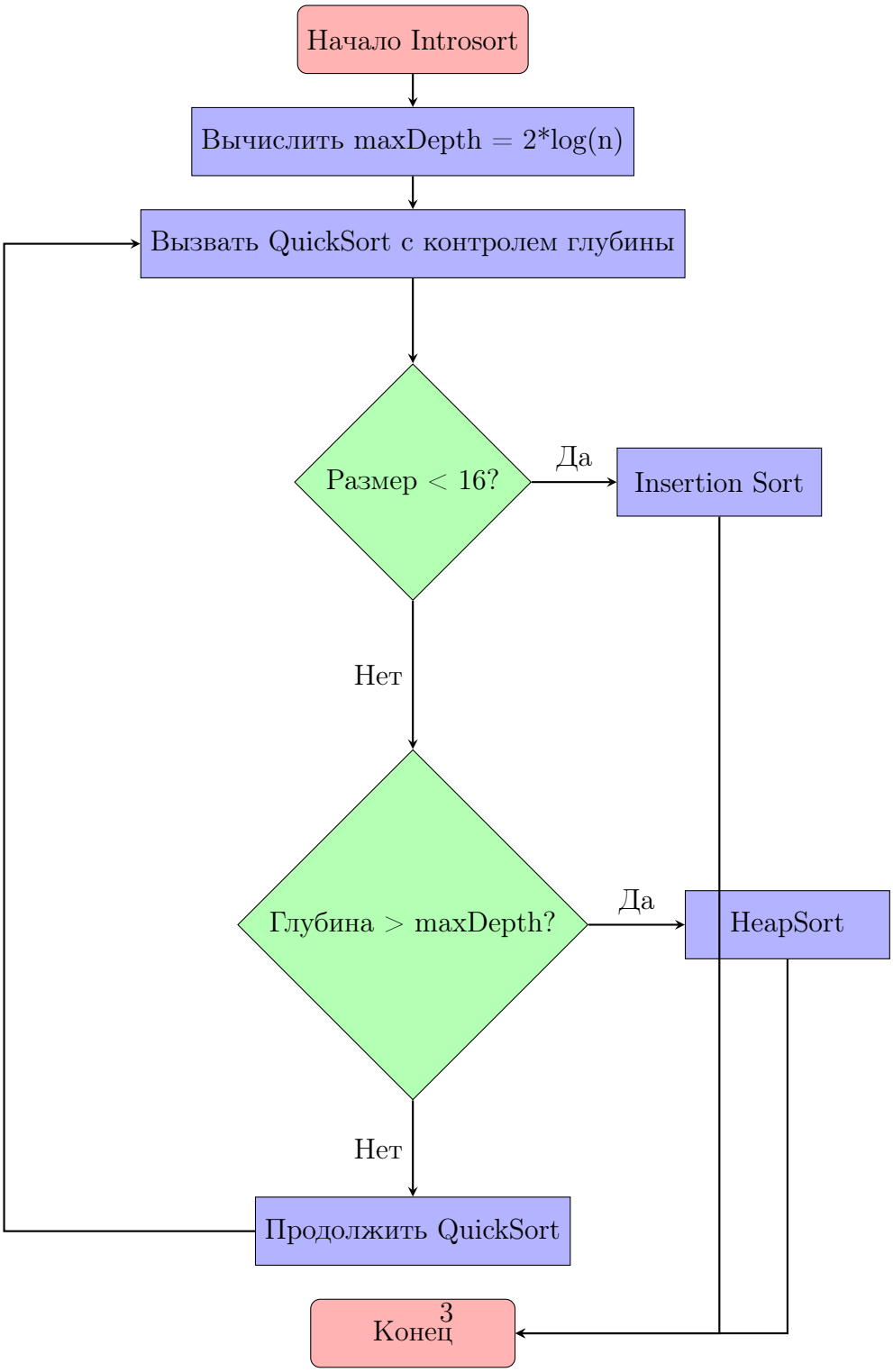
Шаг 2: **Быстрая сортировка:** Рекурсивно разбиваем массив

Шаг 3: **Контроль глубины:** Следим за глубиной рекурсии

Шаг 4: **Переключение:** Если глубина превысила порог $\rightarrow$ переходим на HeapSort

Шаг 5: **Оптимизация:** Для маленьких подмассивов используем Insertion Sort

Блок-схема алгоритма



Пример работы

Массив: [9, 3, 7, 1, 8, 2, 5, 4, 6] **maxDepth = 2*log(9) 6**

1. **Быстрая сортировка:** Выбираем медиану (6), разбиваем:

   ```
   [3, 1, 2, 5, 4] 6 [9, 7, 8]
   Глубина: 1
   ```

2. **Рекурсия:** Обрабатываем левую часть [3, 1, 2, 5, 4]:

   ```
   [1, 2] 3 [5, 4]
   Глубина: 2
   ```

3. **Маленькие подмассивы:** [1, 2] и [5, 4] сортируем Insertion Sort
4. **Правая часть:** [9, 7, 8] - тоже маленькая, Insertion Sort
5. **Результат:** [1, 2, 3, 4, 5, 6, 7, 8, 9]

Пример срабатывания защиты

"Плохой"массив: [1, 2, 3, 4, 5, 6, 7, 8, 9] (уже отсортирован)

1. **Быстрая сортировка:** Выбираем первый элемент (1):

   ```
   [] 1 [2, 3, 4, 5, 6, 7, 8, 9]
   Глубина: 1
   ```

2. **Рекурсия:** Обрабатываем правую часть [2, 3, 4, 5, 6, 7, 8, 9]:

   ```
   [] 2 [3, 4, 5, 6, 7, 8, 9]
   Глубина: 2
   ```

3. **Глубина растет:** Продолжаем до глубины 6...
4. **Срабатывание защиты:** При глубине > maxDepth:

   ```
   Переключаемся на HeapSort для оставшейся части!
   ```

Анализ сложности

- **Лучший случай:** O(n log n) - как QuickSort
- **Средний случай:** O(n log n) - как QuickSort
- **Худший случай:** O(n log n) - благодаря HeapSort!
- **Память:** O(log n) - глубина рекурсии ограничена