Семинар 4

Порядковые статистики. Двоичная куча

Задача – найти *i* -ый по счету минимальный элемент в массиве A
(i-ую порядковую статистику)

Пример:

6	8	9	2

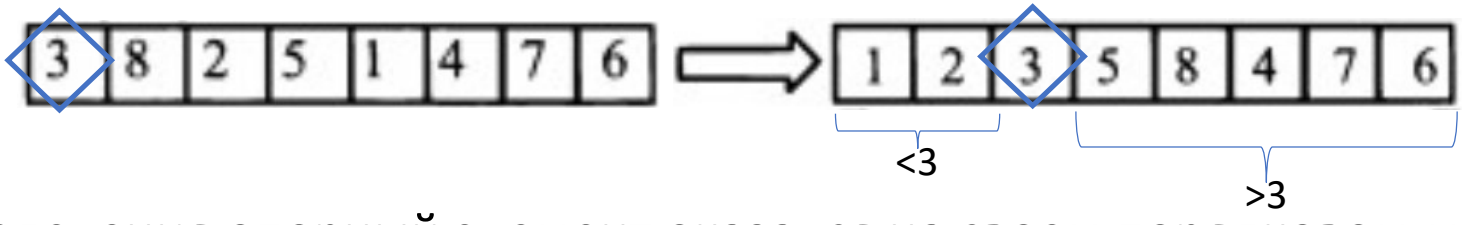
Если *i*=2, то 2-ая статистика равна 6

Если *i*=3, то 3-я статистика равна 8 и т.д.

При *i*=1 ищем min, при *i*=*n* ищем max. Для этих случаев сложность O(n).

Как насчет медианы? Можно отсортировать массив за O(n log n) и выдать любой *i*-ый элемент за O(1). Можно быстрее!

Используем идеи QuickSort (опорные элементы)



После разделения опорный элемент оказался на своем порядковом месте. Если нужна *i*=3-я порядковая статистика, то мы нашли ее. Если *i*>3, то далее обращаемся **только** к правой части, иначе – **только** к левой.

Тестовое задание 1

Ищем 5-ую порядковую статистику. A[1 .. 10].

Пусть после разделения опорный элемент оказался в 3–ей позиции. На какой стороне опорного элемента надо выполнять рекурсию и какую порядковую статистику искать?

a) 3-ю пор. статистику на левой стороне от опорного.

b) 2-ую пор. статистику на правой стороне от опорного.

c) 5-ую пор. Статистику на правой стороне от опорного.

d) Возможно, понадобится рекурсия как слева, так и справа от опорного.

RSELECT

Вход: массив A из $n \geq 1$ разных чисел и целое число $i \in \{1, 2, ..., n\}$.

Выход: i-я порядковая статистика массива A.

```
if n = 1 then                                  // базовый случай
    return A[1]
выбрать опорный элемент p равномерно случайным образом из A
разделить A вокруг p
j := позиция p в разделенном массиве
if j = i then                                  // вам посчастливилось!
    return p
else if j > i then
    return RSelect(first part of A, i)
else                                           // j < i
    return RSelect(second part of A, i - j)
```

Можно доказать, что время работы RSELECT равно в **среднем** O(n) $T_{\text{ср}}(n) \leq T_{\text{ср}}\left(\frac{n}{2}\right) + O(n) \Rightarrow O(n)$

Вопрос: может ли в процедуре RSELECT в качестве параметра передаваться массив с нулевым количеством элементов?

Алгоритм DSELECT

Сначала рекурсивно находим медиану (медиану медиан). Потом производим разделение массива вокруг нее и ищем порядковую статистику рекурсивно как в RSELECT.

Нахождение медианы медиан (пример)

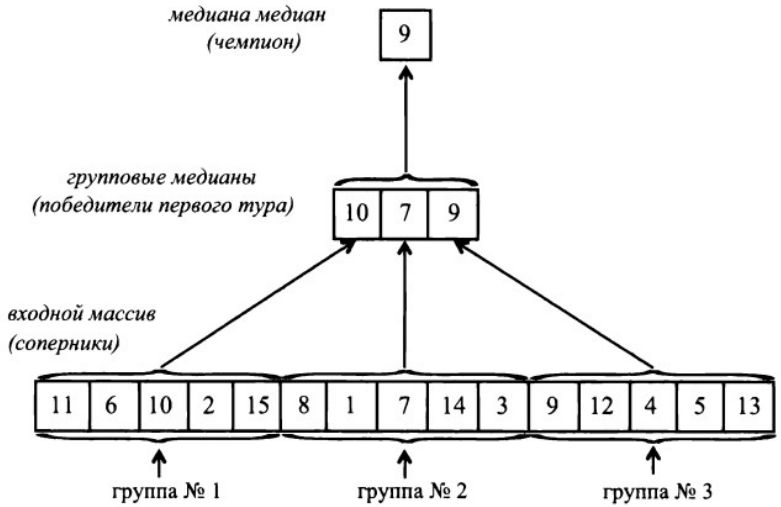


Рис. 6.1. Вычисление опорного элемента на основе турнира на выбывание в два тура. В этом примере выбранный опорный элемент — это не медиана входного массива, а довольно близкое к ней значение

Сложность O(n)

Двоичная куча

Структуры данных позволяют так организовать данные, что получать к ним доступ быстро и с пользой.

- очереди
- стеки
- кучи
- двоичные деревья
- хеш-таблицы
- фильтры Блума
- union-find

Для разных задач выбираются различные структуры данных

Куча отслеживает эволюционирующее множество объектов и может быстро идентифицировать объект с наименьшим (наибольшим)ключом.

Основные кучевые операции:

- Вставить (элемент);
- Извлечь минимум.

Вставить элемент в массив можно за O(1), извлечь минимальный за O(n).

Если массив отсортирован, вставить можно за O(n), зато извлечь минимальный за O(1).

Кучевые операции	**Время**
Вставить	$O(\log n)$
Извлечь минимум	$O(\log n)$
Найти минимум	$O(1)$
Объединить в кучу	$O(n)$
Удалить из кучи	$O(\log n)$

Задачи с применением кучи

1. Сортировка

HEAPSORT (КУЧЕВАЯ СОРТИРОВКА)

Вход: массив A из n несовпадающих целых чисел.

Выход: массив B с теми же целыми числами, отсортированными от наименьшего к наибольшему.

H := пустая куча

for i = 1 to n **do**

  Вставить $A[i]$ в H

for i = 1 to n **do**

  $B[i]$:= Извлечь минимум из H

Построить кучу за $O(n \log n)$или за $O(n)$, затем последовательно извлекать min

Тестовое задание 2

Каково время работы алгоритма `HeapSort` как функции от длины n входного массива?

а) $O(n)$

б) $O(n \log n)$

в) $O(n^2)$

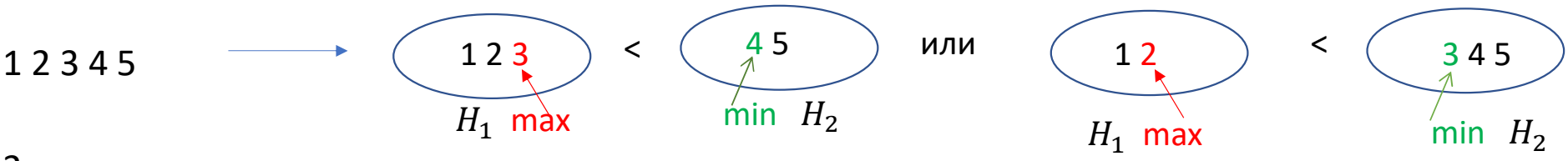
г) $O(n^2 \log n)$

2. Событийный менеджер. Например, разработка баскетбольной видеоигры. Необходимо следить за событиями и постоянно определять, что произойдет дальше. Это сводится к повторным вычислениям min на множестве запланированных событий. События хранятся в куче с ключами, равными запланированному времени. Новые события вставляется в кучу по мере их возникновения.

3. Поддержка медианы.

Есть массив из (разных) чисел. При получении нового числа ответить медианой.

Будем поддерживать 2 кучи (H_1 и H_2) для меньшей и большей половин массива.



2 **инварианта:**

- кучи сбалансированы
- кучи упорядочены

Медианный элемент 3

Поддержка медианы (продолжение)

1 2 3 4 5 6 → (1 2 3) < (4 5 6)

(1 2 3): H_1, max

(4 5 6): min, H_2

Медианные элементы 3 и 4

Для H_1 кучевые операции «вставить» и «извлечь max»

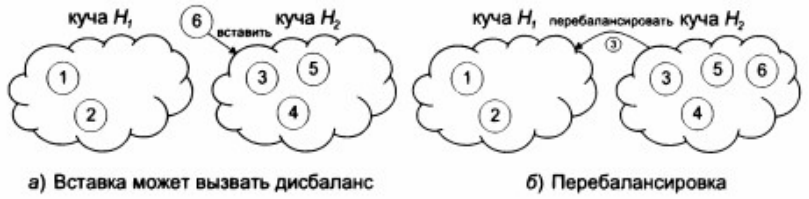
Для H_2 кучевые операции «вставить» и «извлечь min»

Как поддерживать
2 **инварианта?**
-кучи сбалансированы
-кучи упорядочены

Вставить новый элемент x (куда – в H_1 или в H_2)?

Если $x < max(H_1)$, то в H_1,
если $x > min(H_2)$, то в H_2,
Если $max(H_1) < x < min(H_2)$ – в любую кучу

Четный раунд



а) Вставка может вызвать дисбаланс

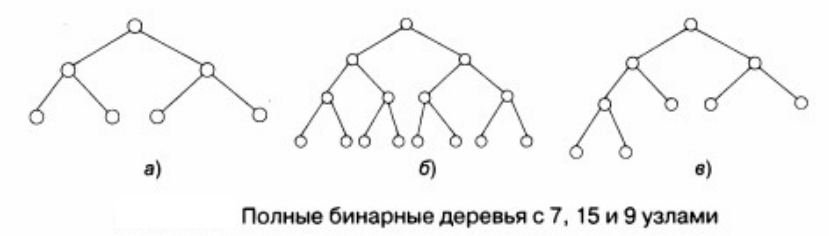
б) Перебалансировка

При вставке нового элемента куча H_2 содержит на два элемента больше, чем куча H_1, наименьший элемент в H_2 извлекается и повторно вставляется в H_1 для восстановления баланса

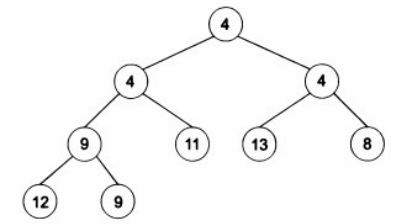
Какова сложность вставки?

Реализация кучи. Куча управляет объектами, ассоциированными с ключами, чтобы соблюдалось свойство кучи: для каждого объекта x ключ объекта меньше или равен ключам его потомков)

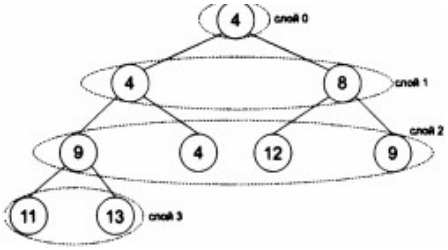
1. В виде дерева (полного бинарного дерева)



Полные бинарные деревья с 7, 15 и 9 узлами



или



ключ объекта меньше или равен ключам его потомков

2. В виде массива

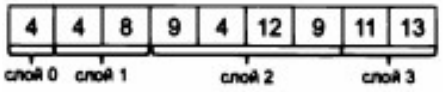
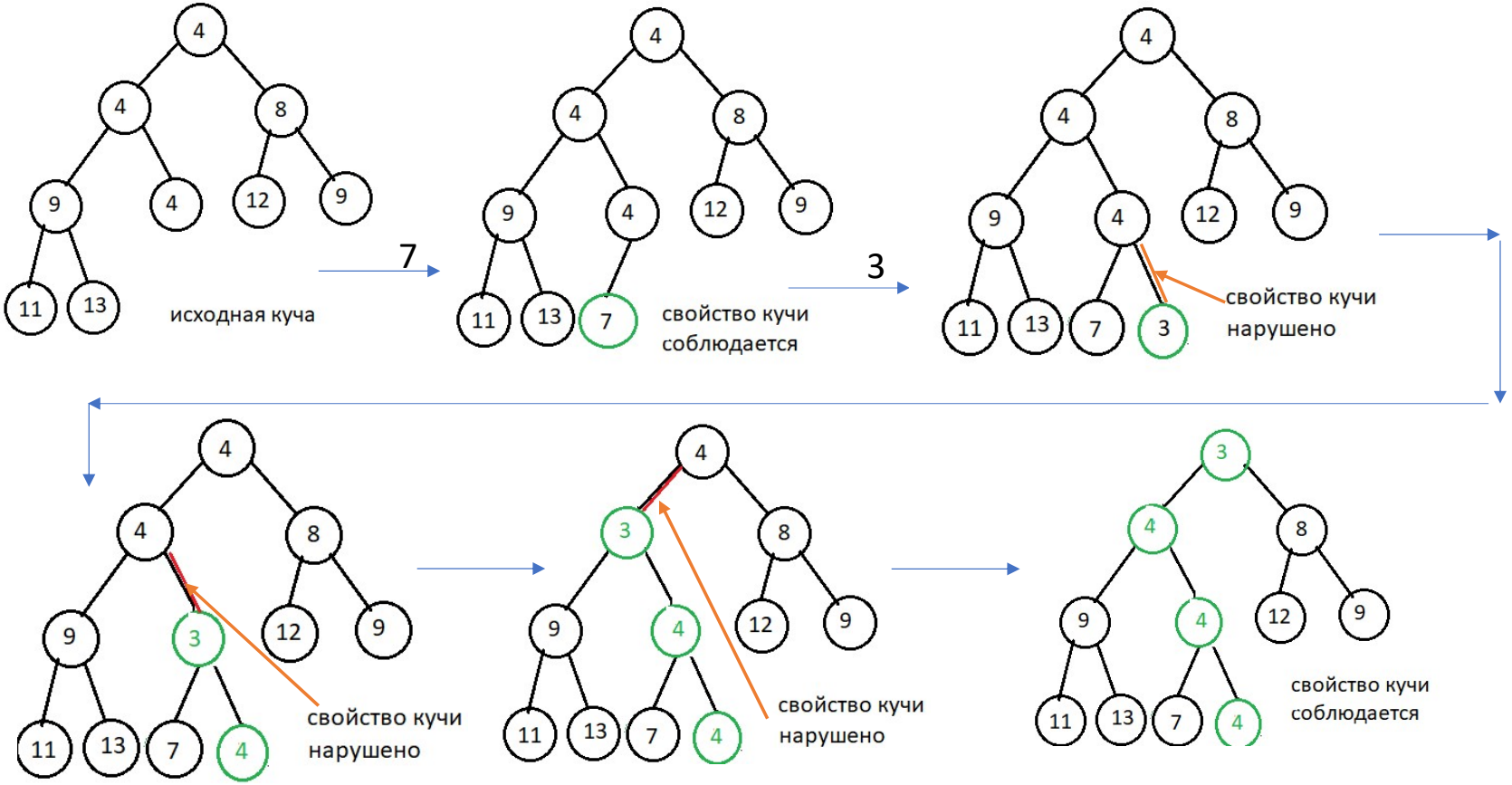


Таблица Взаимосвязи между позицией $i \in \{1, 2, 3, \ldots, n\}$ объекта в куче и позициями его родителя, левого потомка и правого потомка, где n обозначает число объектов в куче

Позиция родителя	$\lfloor i/2 \rfloor$ (при условии, что $i \geq 2$)
Позиция левого потомка	$2i$ (при условии, что $2i \leq n$)
Позиция правого потомка	$2i + 1$ (при условии, что $2i + 1 \leq n$)

Операция «вставить» (объект в кучу)

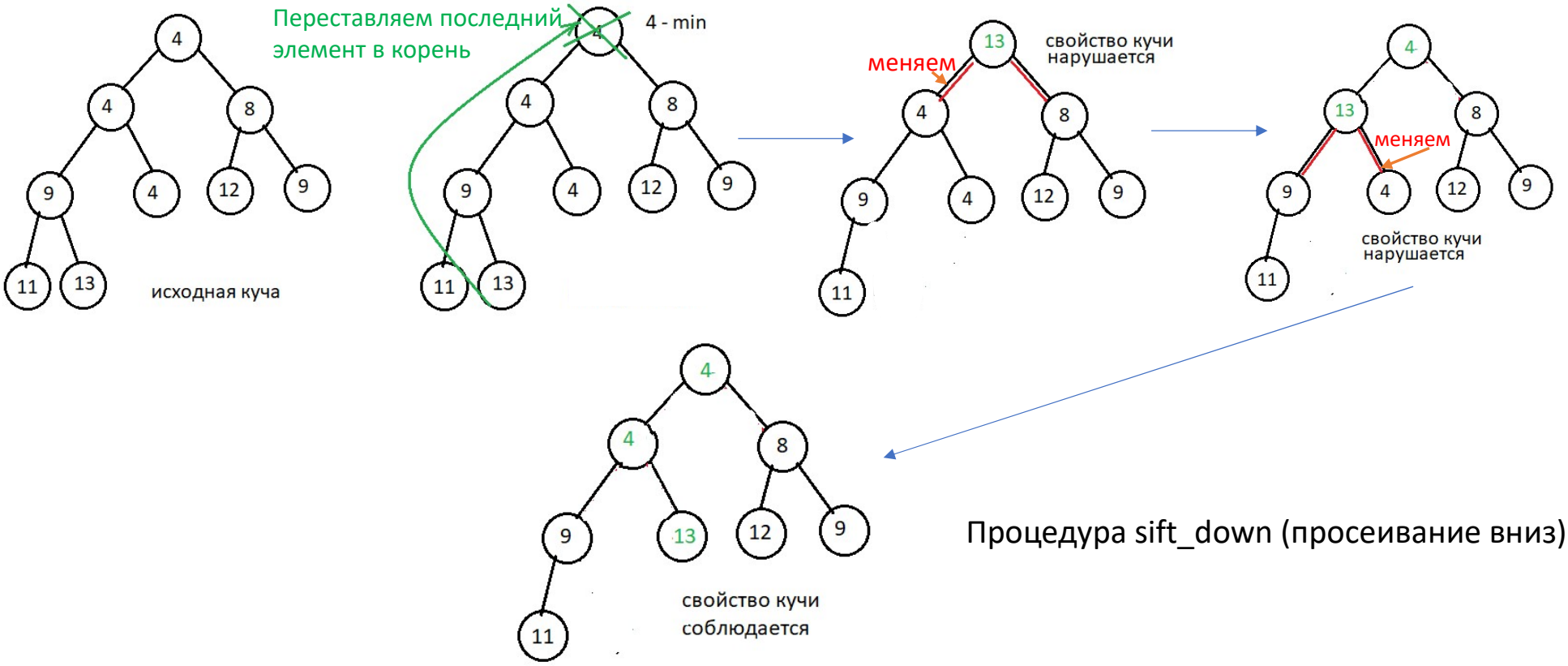


Сложность?

Процедура shiftup (просеивание вверх)

Написать процедуру sift_up

Операция «извлечь минимум»



Процедура sift_down (просеивание вниз)