Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет телекоммуникаций и

информатики»

(СибГУТИ)

02.03.02 Фундаментальная информатика и информационные технологии

Профиль: Системное программное обеспечение

(очная форма обучения)

ОТЧЕТ ПО учебной ПРАКТИКЕ

в/на

(наименование профильной организации/структурного подразделения СибГУТИ)

ТЕМА ИНДИВИДУАЛЬНОГО ЗАДАНИЯ

Выполнил:

студент института ИВТ

гр. ИВ-021 / /

«27» мая 2023г. (подпись)

Проверил:

Руководитель от СибГУТИ / /

«27» мая 2023г. (подпись)

Новосибирск 2023

**План-график проведения**  учебной\_\_\_\_\_\_ **практики**

Вид практики

Фамилия Имя Отчество студента

института Информатика и вычислительная техника , 2 курса, гр. ИС-021

Направление: 02.03.02 Фундаментальная информатика и информационные технологии

Код – Наименование направления (специальности)

Профиль: Системное программное обеспечение

Место прохождения практики

Объем практики: **108/3** часов/ЗЕ

Вид практики  ***учебная***

Тип практики ***научно-исследовательская работа (получение первичных навыков научно-исследовательской работы)***

Срок практики с "30" января 2023 г.

по "27" мая 2023 г.

Содержание практики\*:

|  |  |
| --- | --- |
| Наименование видов деятельности | Дата (начало – окончание) |
| 1. Общее ознакомление со структурным подразделением предприятия, вводный инструктаж по технике безопасности | 30.01.2023–01.02.2023 |
| 2. Выдача задания на практику, деление студентов на группы (если необходимо), определение конкретной индивидуальной темы, формирование плана работ | 02.02.2023–04.02.2023 |
| 3. Работа с библиотечными фондами структурного подразделения или предприятия, сбор и анализ материалов по теме практики | 06.02.2023–11.02.2023 |
| 4. Выполнение работ в соответствии с составленным планом:  –  –  –  (перечисление конкретных видов работ, связанных с выполнением поставленных задач) | 13.02.2023 – 20.05.2023 |
| 5. Анализ полученных результатов и произведенной работы  Составление отчета по практике, защита отчета | 22.05.2023–27.05.2023 |

\*В соответствии с программой практики

Руководитель от СибГУТИ / /

«28» \_\_\_\_01\_\_\_\_ 2023г. (подпись)

**ЗАДАНИЕ НА ПРАКТИКУ**

**ВВЕДЕНИЕ**

В первую очередь нужно разобраться, что из себя представляют деревья, ведь бинарная куча – это бинарное дерево. Корневое дерево – это структура данных, которая представляет собой совокупность связанных узлов, один из которых является корнем дерева. Ребра связывают узлы дерева и устанавливают между ними отношение «родитель-дочерний».

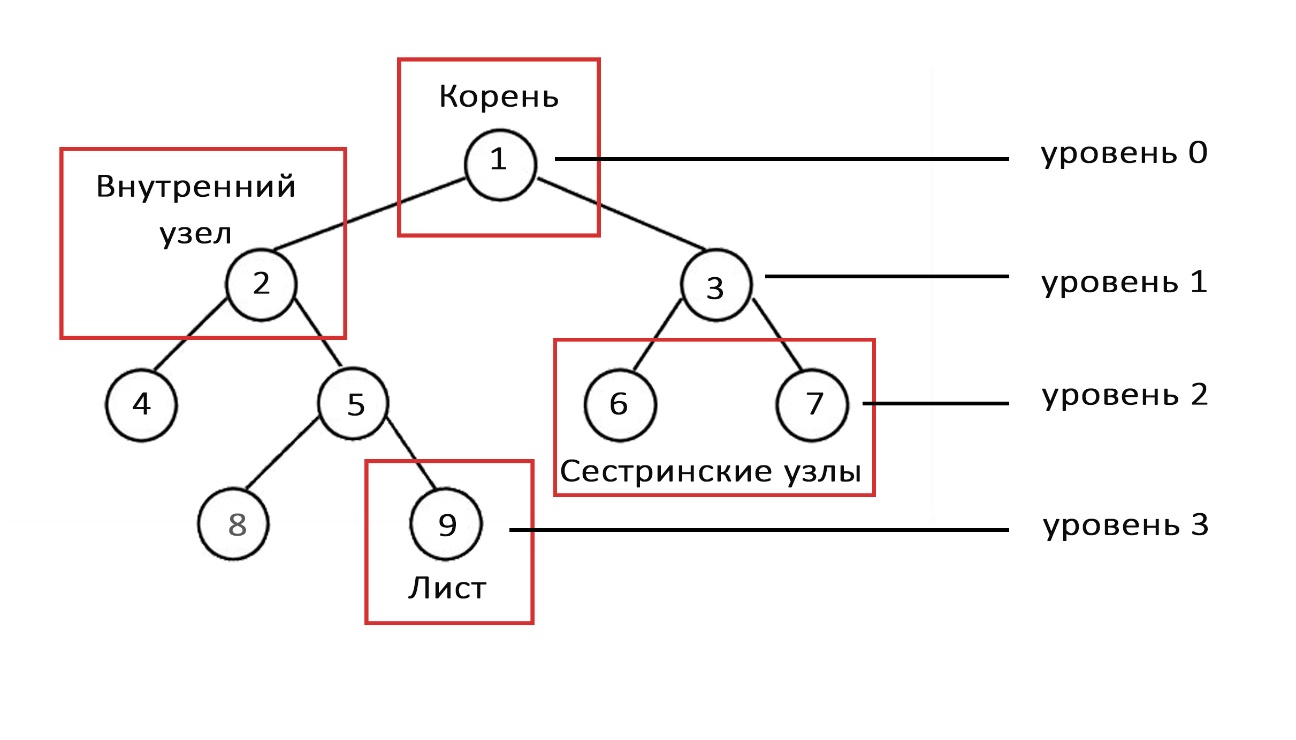


Рис. 1. Пример корневого дерева.

Узел, который не имеет дочерних вершин, называется внешним узлом, или листом. Аналогично, узел дерева, имеющий дочерние вершины называется внутренним узлом. Любой внутренний узел по отношению к своим дочерним вершинам называется родительским. Узлы, имеющие общего родителя, называются родственными, или сестринскими узлами. Любой узел в поддереве некоторой вершины называется ее потомком, а вершина для этих узлов является предком. Степень узла – это количество его дочерних узлов. Высота узла – количество ребер в длиннейшем пути от узла до листа. Высота дерева – количество ребер в длиннейшем пути от корня до листа. Глубина узла или уровень узла – количество ребер в пути от узла до корня.

Бинарное дерево (двоичное дерево) – это корневое дерево, в котором каждый узел имеет не более двух дочерних вершин (0, 1 или 2 вершины).

Полное бинарное дерево – это бинарное дерево, в котором каждый узел имеет 0 или 2 дочерних узла. Пример такого дерева приведен на рис. 1.

Завершенное бинарное дерево – это бинарное дерево, в котором каждый уровень, возможно за исключением последнего, полностью заполнен узлами, а заполнение последнего уровня осуществляется слева направо.

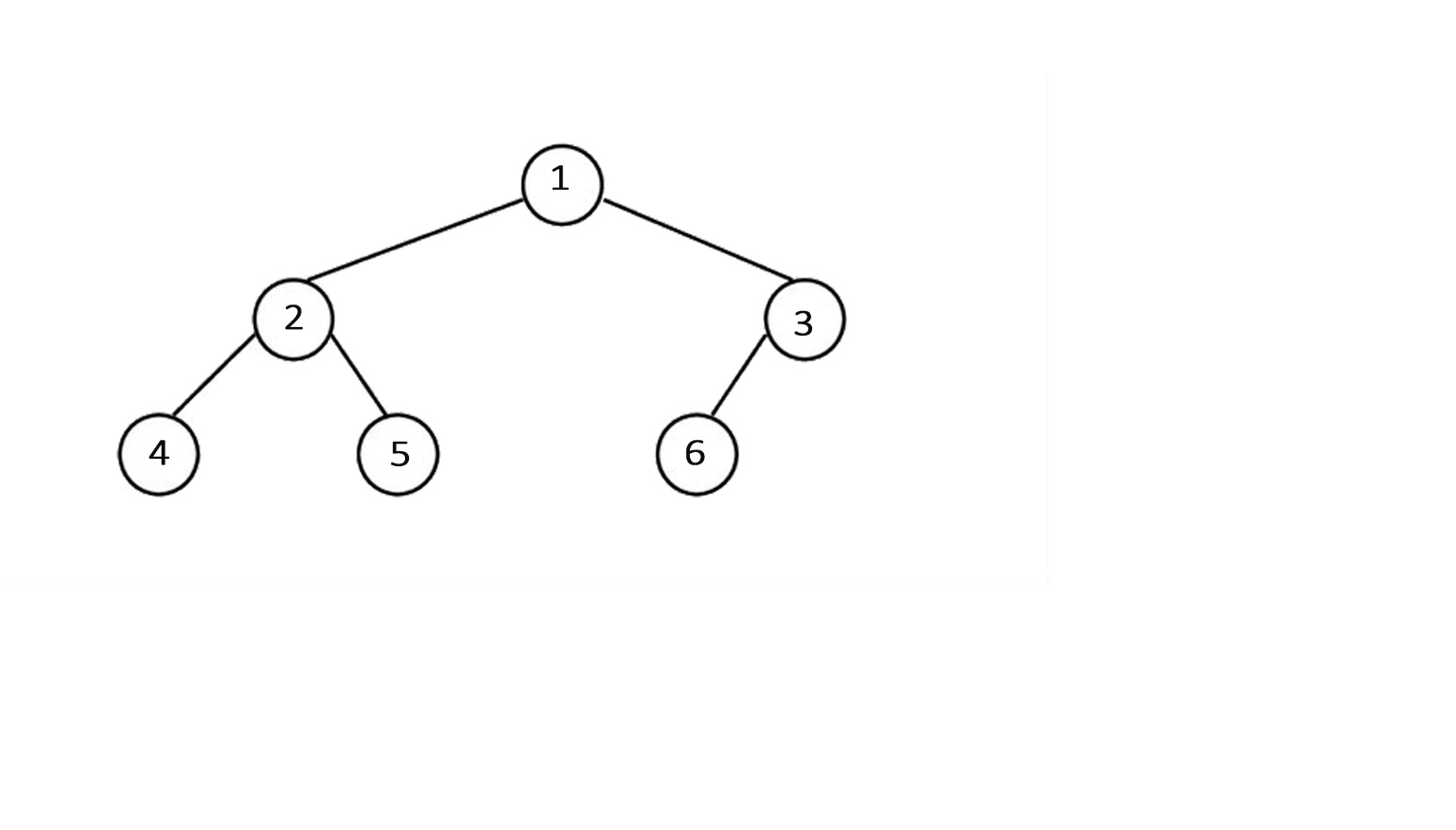


Рис. 2. Завершенное двоичное дерево.

Совершенное двоичное дерево — особый тип бинарного дерева, в котором у каждого внутреннего узла по два ребенка, а листовые вершины находятся на одном уровне (рис. 3).

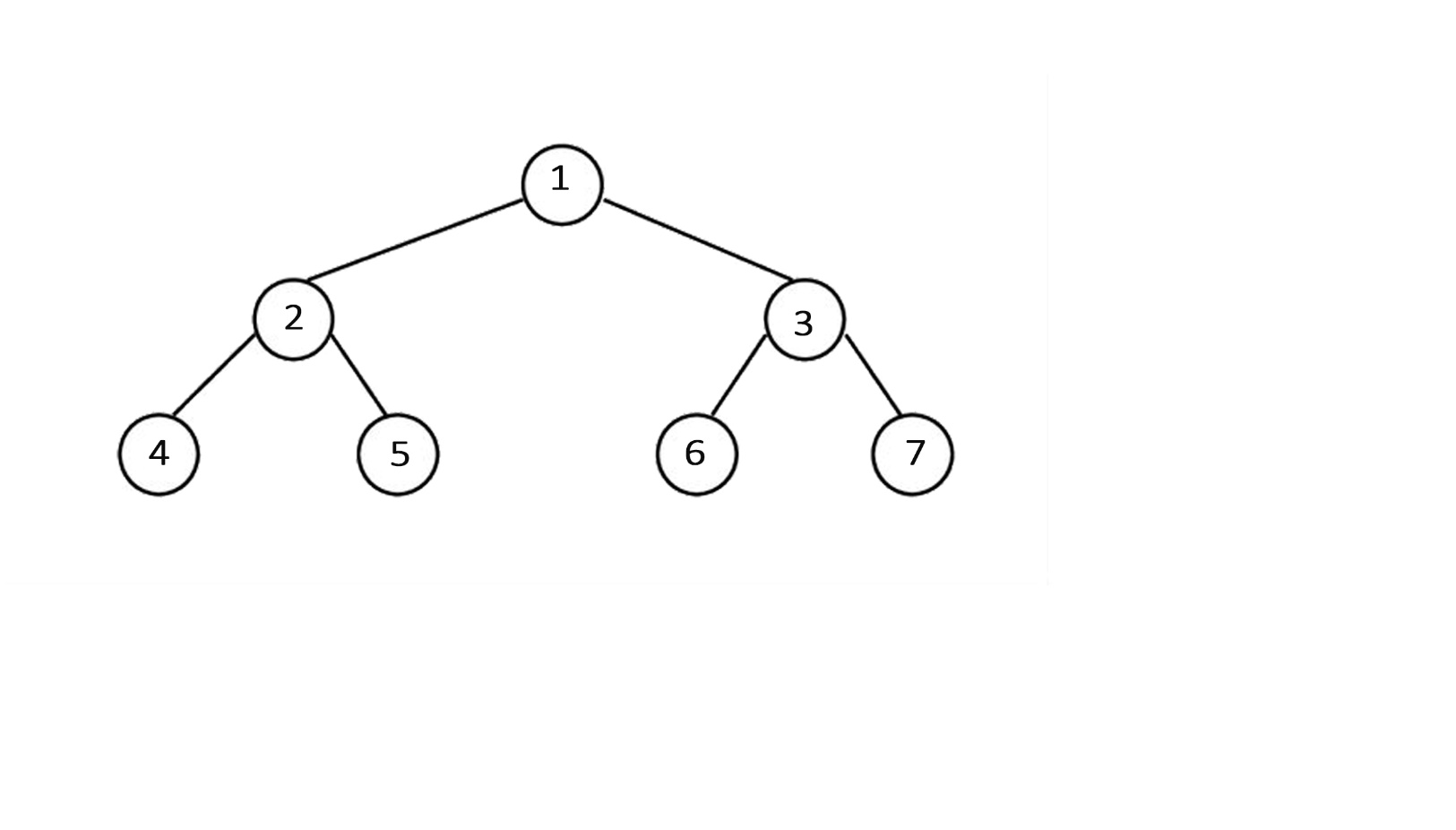


Рис. 3. Совершенное двоичное дерево

Теперь разберёмся, что такое бинарная куча. Бинарной кучей (binary heap) называется завершенное бинарное дерево, в котором каждый узел содержит ключ и значение. Ключ выступает в роли приоритета узла, а значение – это некоторые данные, ассоциированные с ключом. Структура дерева и распределение ключей по нему удовлетворяют следующим требованиям:

* Каждый уровень дерева, возможно за исключением последнего, полностью заполнен узлами, а заполнение последнего уровня осуществляется слева направо
* Ключ любого узла не меньше (не больше) ключей его потомков

Бинарные кучи (binary heap) предназначены для реализации АТД очередь с приоритетом. Они поддерживают операции поиска и удаления элемента с экстремальным значением ключа (либо минимальным, либо максимальным), но не предоставляют возможности поиска элемента по ключу.

Бинарные кучи также называют пирамидами и сортирующими деревьями. Алгоритмы этой структуры данных лежат в основе пирамидальной и турнирной сортировки.

Число уровней в завершенном бинарной куче из 𝑛 ключей равно ⌊log2 𝑛⌋ + 1.

Бинарная куча, в которой ключ любого узла не меньше ключей его потомков, называется невозрастающей бинарной кучей, или невозрастающей пирамидой (max-heap). В таком дереве ключ с максимальным значением расположен в его корне.

Аналогично, бинарная куча, в которой ключ любого узла не больше ключей его потомков, называется неубывающей бинарной кучей, или неубывающей пирамидой (min-heap). В корне дерева находится ключ с минимальным значением.

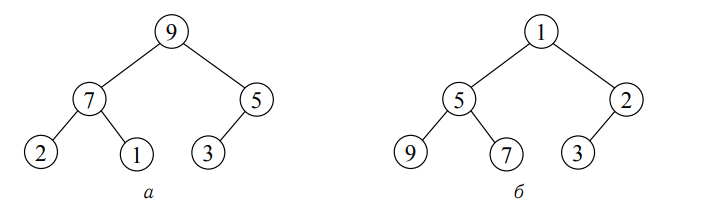


Рис. a - невозрастающая куча (max-heap);

б – неубывающая куча (min-heap).

Представление бинарной кучи в памяти.

Завершенные бинарные деревья можно хранить в одномерном массиве. Все узлы дерева размещаются в ячейках массива 𝐴[0..𝑛 - 1] в порядке их обхода в ширину. Корень дерева хранится в первой ячейке массива. Для любой вершины, размещенной в ячейке 𝑖 ∈ {1, 2, . . . , 𝑛 - 1}, левый дочерний узел LEFT(𝑖) расположен в позиции 2𝑖 + 1, а правый дочерний узел RIGHT(𝑖) в позиции 2𝑖 + 2. Родительский узел PARENT(𝑖) вершины 𝑖 находится в ячейке ⌊(𝑖 – 1) /2⌋. Если значения 2𝑖 + 1 или 2𝑖 + 2 больше 𝑛, то у вершины 𝑖 отсутствует соответствующий дочерний узел.

Для хранения элементов бинарной кучи используется массив 𝐴[0..𝑚-1] из 𝑚 ячеек. Каждая ячейка 𝑖 хранит ключ 𝐴[𝑖].𝑘𝑒𝑦 и некоторое значение 𝐴[𝑖].𝑣𝑎𝑙𝑢𝑒. Очевидно, что при таком представлении максимальное число узлов в куче не превышает 𝑚. Для определенности обозначим через 𝑛 количество элементов, реально хранящихся в бинарной куче (𝑛 ≤ 𝑚).

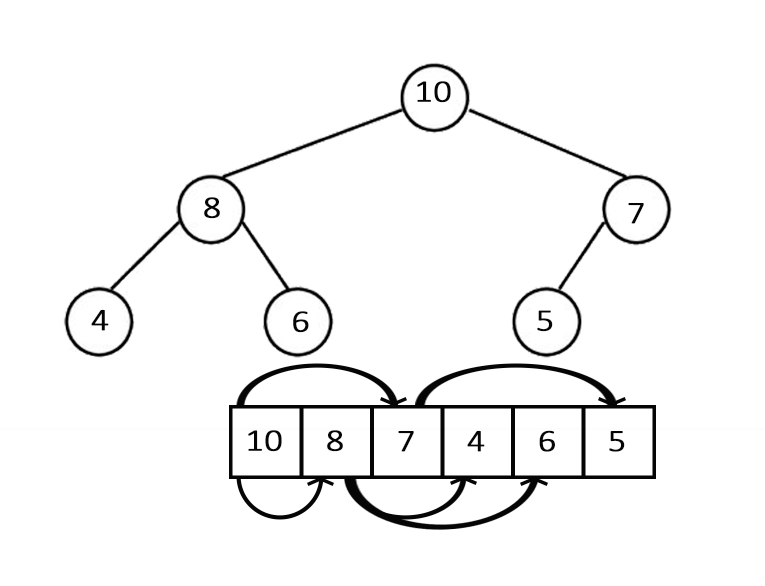


Рис. Представление бинарной кучи из 6 элементов (max-heap)

в виде массива 𝐴[0..5]

**Реализация операций над бинарной кучей.**

Рассмотрим реализации операций невозрастающей бинарной кучи (max-heap). Операции неубывающей бинарной кучи реализуются аналогично.

Структура узлов кучи:

struct binary\_heap\_nodes {

int key;

int value;

};

Структура бинарной кучи:

struct binary\_heap {

int n; // число элементов

int m; // размер массива n <= m

struct binary\_heap\_nodes \*nodes;

};

**Создание кучи.** Для инициализации кучи вызывается функция **heap\_init**. При инициализации задаётся значение m, зануляется n и выделяется динамическая память под массив узлов.

**Вставка элемента.** Новый элемент добавляется в конец массива. В дереве этой позиции соответствует самый правый лист на последнем уровне. Если свободных ячеек в массиве не осталось, функция **heap\_insert** возвращает специальное значение HEAP\_OVERFLOW.

После вставки нового элемента могут быть нарушены свойства невозрастающей бинарной кучи – добавленный ключ может быть больше ключа родительского узла. Поэтому нам необходимо восстановить свойства бинарной кучи. Для этого мы проходим в функции **heapifyup** по дереву от листа 𝐴[𝑛-1] до корня 𝐴[0] и выполняем обмены узлов, если ключ родительского узла меньше ключа текущего элемента. Перебор узлов прерывается, если ключ родителя стал больше ключа текущего узла.

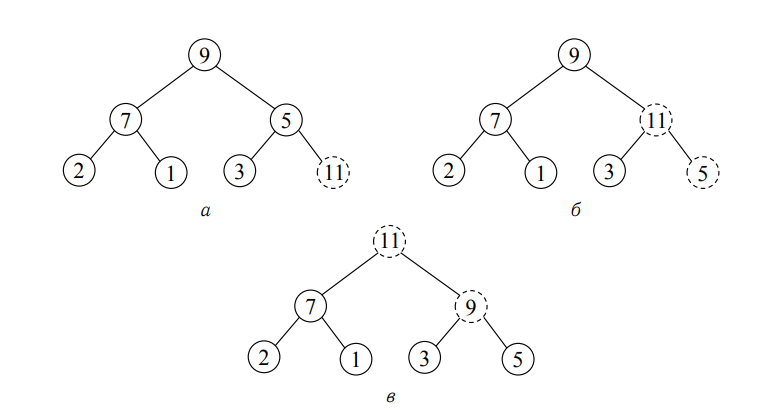


Рис. Добавление ключа 11 в невозрастающую бинарную кучу: а – добавление ключа 11 в конец массива – ячейку 𝐴[6]; б – результат обмена значениями узлов 11 и 5; в – результат обмена значениями узлов 11 и 9.

**Поиск максимального элемента.** По определению элемент с максимальный ключом (приоритетом) хранится в первой ячейке массиве. Если куча не содержит элементов, то функция **heap\_max** возвращает специальное значение HEAP\_EMPTY.

**Удаление максимального элемента.** Для удаления максимального элемента 𝐴[0] мы запоминаем его значение для последующего возврата из функции **heap\_delete\_max**. Далее последний элемент 𝐴[𝑛-1] переносится в ячейку 𝐴[0], число 𝑛 элементов в куче уменьшается на 1. После этого необходимо восстановить свойства невозрастающей кучи, так как в корне дерева теперь может находиться элемент не с максимальным значением.

Восстановление свойств кучи выполняется функцией **heapifydown**, которая в цикле движется от корня к листьям. На каждой итерации цикла среди текущего узла 𝑖 и двух его дочерних элементов 𝑙𝑒𝑓 𝑡 и 𝑟𝑖𝑔ℎ𝑡 выбирается наибольший 𝑙𝑎𝑟𝑔𝑒𝑠𝑡 – тот, который содержит наибольший ключ. Если наибольшим оказался один из дочерних узлов, то выполняется обмен текущего узла 𝑖 и найденного наибольшего. Проход по дереву останавливается, если наибольшее значение ключа оказалось у текущего элемента 𝑖, а не у его дочерних. Это означает, что дальше свойства кучи восстанавливать не требуется.

Время выполнения функции **heapifydown** зависит от высоты ℎ узла 𝑖, для которого она вызвана. Таким образом, время выполнения этой функции есть 𝑂(ℎ). В худшем случае **heapifydown** осуществляет проход по всем уровням завершенного бинарного дерева (от корня до листа), что требует 𝑂(log 𝑛) операций обменов.

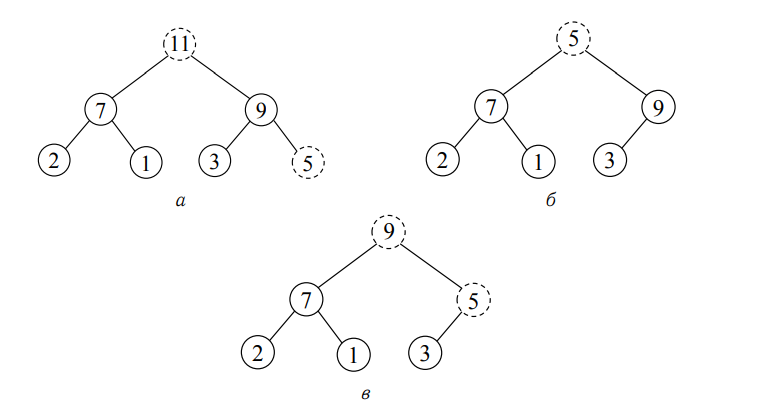


Рис. 14.4. Удаление ключа 11 с максимальным значением (max-heap): а – исходное состояние кучи; б – последний элемент 5 массива перемещен в корень дерева; в – результат обмена значениями узлов 5 и 9.

**Увеличение ключа**. Операция heap\_increase\_key заменяет у узла, хранящегося в ячейке 𝑖, значение ключа на большее. Если переданное значение нового ключа 𝑘𝑒𝑦 меньше текущего 𝐴[𝑖].𝑘𝑒𝑦, то функция возвращает специальное значение HEAP\_INVALID\_KEY.

После установки нового значения ключа следует восстановить свойства невозрастающей кучи. Новый ключ может быть больше ключа родительского элемента, поэтому мы вызываем функцию **heapifyup** для ячейки с номером 𝑖.

В худшем случае выполняется изменение приоритета листового узла и новый ключ превосходит по значению все имеющиеся ключи. В такой ситуации функция **heapifyup** выполняет порядка 𝑂(log 𝑛) обменов узлов.

**Построение кучи.** Если нам заранее задан массив 𝐴[0..𝑚-1], содержащий 𝑛 ключей, то мы можем преобразовать его в невозрастающую бинарную кучу за время 𝑂(𝑛).

Любой массив чисел можно интерпретировать как завершенное бинарное дерево. Чтобы сформировать из него невозрастающую бинарную кучу требуется восстановить свойство распределения ключей. Для этого можно использовать процедуру **heapifydown**. При помощи нее мы расставляем ключи на корректные позиции.

В завершенном бинарном дереве элементы, хранящиеся в ячейках ⌊𝑛/2⌋ + 1, ⌊𝑛/2⌋ + 2, . . . , 𝑛, не имеют дочерних узлов (являются листьями). Поэтому восстановление свойств кучи целесообразно начинать с ячейки ⌊𝑛/2⌋ и двигаться в направлении начала массива (корня дерева).

Очистка памяти. Так как при инициализации кучи выделялась динамическая память, в завершении работы её нужно освободить. Для этого вызывается функция heap\_**free**.

**Скорость операций**

Вставка элементов 𝑂(log 𝑛)

Поиск максимального элемента 𝑂(1)

Увеличение ключа 𝑂(log 𝑛)

Удаление максимального элемента 𝑂(log 𝑛)

Построение кучи 𝑂(𝑛)

**ЗАКЛЮЧЕНИЕ**

в котором анализируется проведенная работа в целом, дальнейшие пути исследований и т.п.;

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

(список литературы, изученной и/или использованной в процессе прохождения практики);

Список литературы должен содержать как минимум один источник из библиотеки СибГУТИ (печатный или электронный). Правила оформления списка литературы см. в ЭИОС https://eios.sibsutis.ru/course/view.php?id=1251

**ПРИЛОЖЕНИЯ**

(если нужно представить результаты выполненной работы более подробно, например, в виде таблиц, графиков, программного кода и т.п.).

**Отзыв о работе студента**

|  |
| --- |
|  |
| (ФИО студента) |
|  |

Уровень освоения компетенций

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | (ФИО студента) | |
|  |  | |
| Компетенции | | Уровень сформированности  компетенций |
| *ОПК-1 - Способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной*  *деятельности* | |  |

отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от СибГУТИ:

Должность руководителя подпись ФИО руководителя

"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.