

Лабораторна робота №9. Пірамідальне сортування

Мета: розробити програмні реалізації алгоритму пірамідального сортування.
Провести аналіз алгоритму

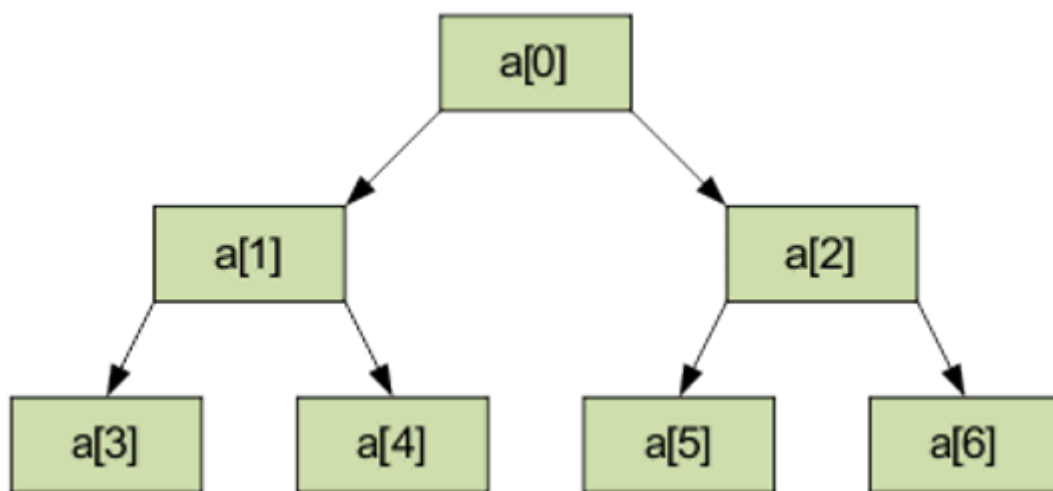
Теоретичні відомості

Метод пірамідальної сортировки, изобретенный Д. Уилльямсом, является улучшением традиционных сортировок с помощью дерева.

Пирамидой (кучей) называется двоичное дерево такое, что

$$a[i] \leq a[2i+1];$$

$$a[i] \leq a[2i+2].$$



$a[0]$ — минимальный элемент пирамиды.

Общая идея пирамидальной сортировки заключается в том, что сначала строится пирамида из элементов исходного массива, а затем осуществляется сортировка элементов.

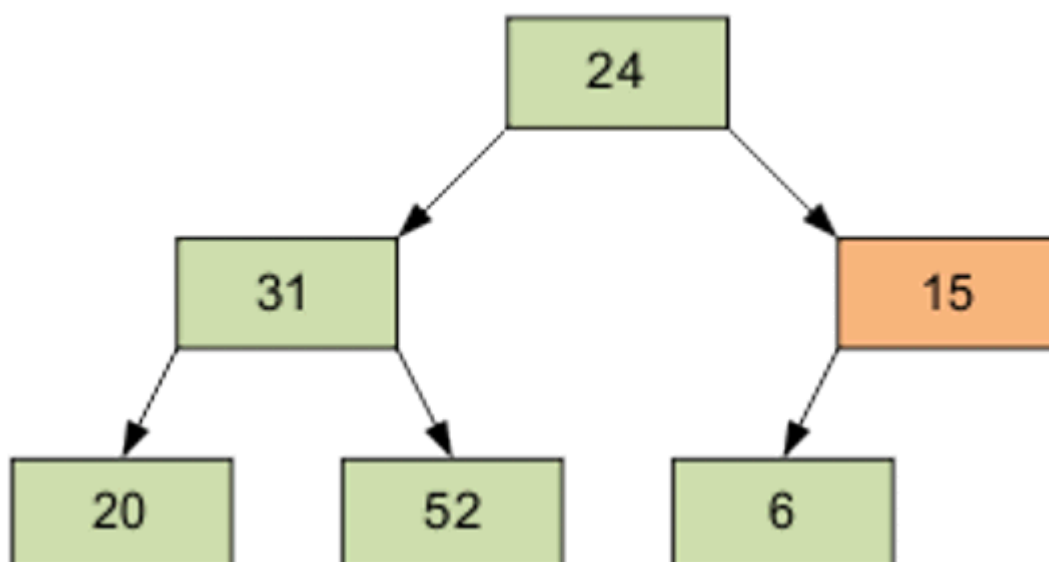
Выполнение алгоритма разбивается на два этапа.

1 этап Построение пирамиды. Определяем правую часть дерева, начиная с $n/2-1$ (нижний уровень дерева). Берем элемент левее этой части массива и просеиваем его сквозь пирамиду по пути, где находятся меньшие его элементы, которые одновременно поднимаются вверх; из двух возможных путей выбираете путь через меньший элемент.

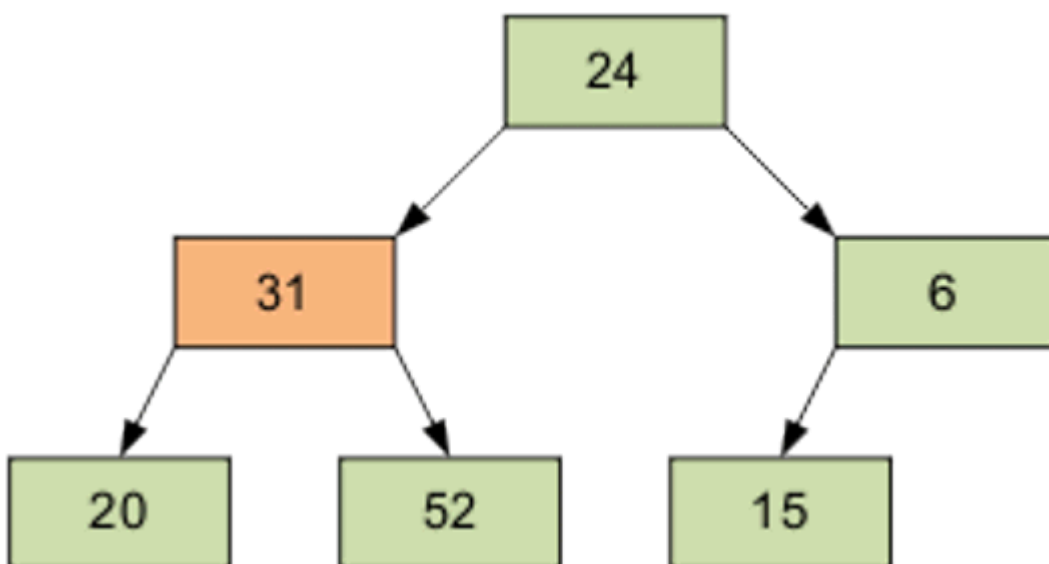
Например, массив для сортировки

24, 31, 15, 20, 52, 6

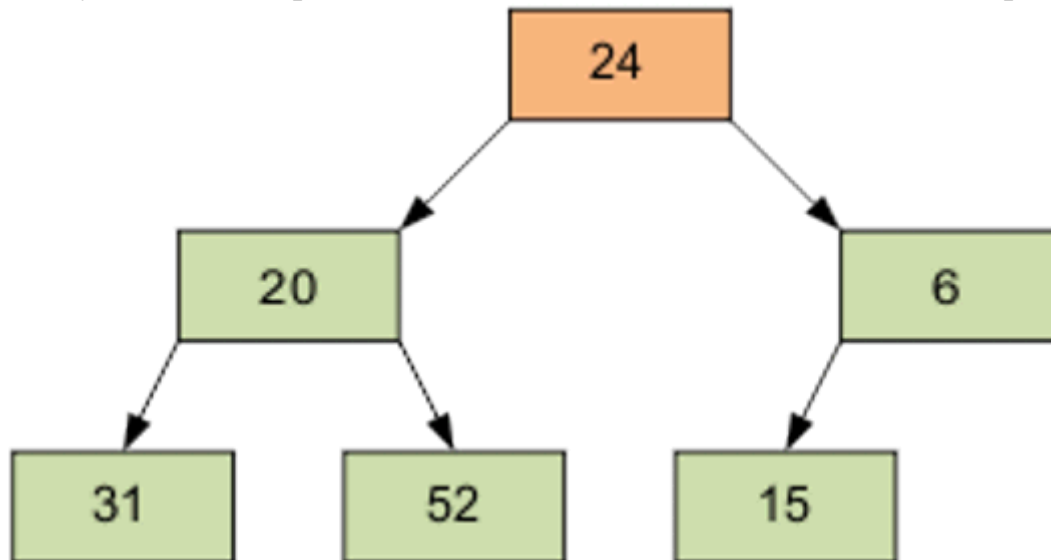
Расположим элементы в виде исходной пирамиды; номер элемента правой части $(6/2-1)=2$ — это элемент 15.



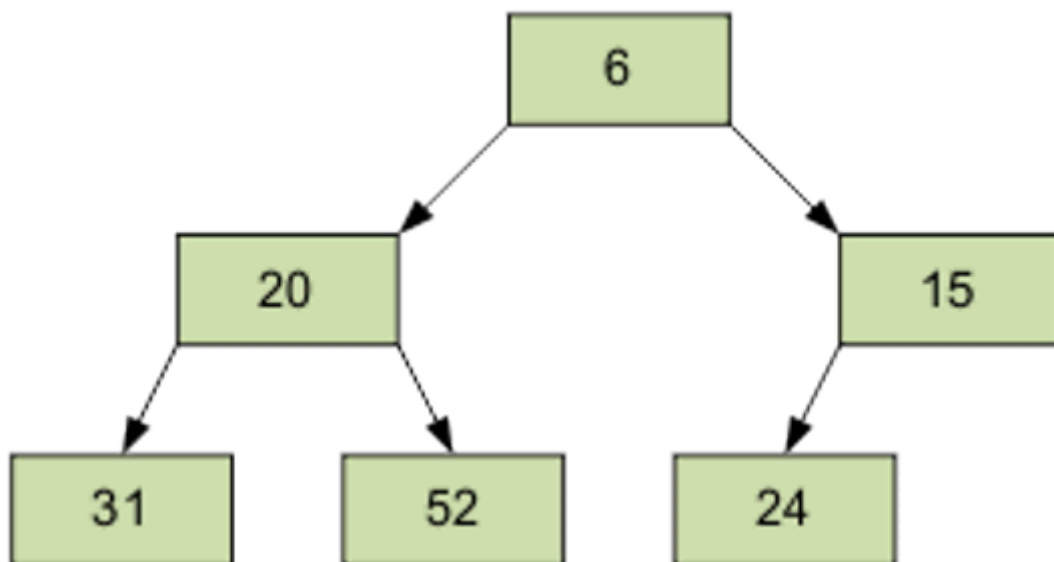
Результат просеивания элемента 15 через пирамиду.



Следующий просеиваемый элемент – 1, равный 31.

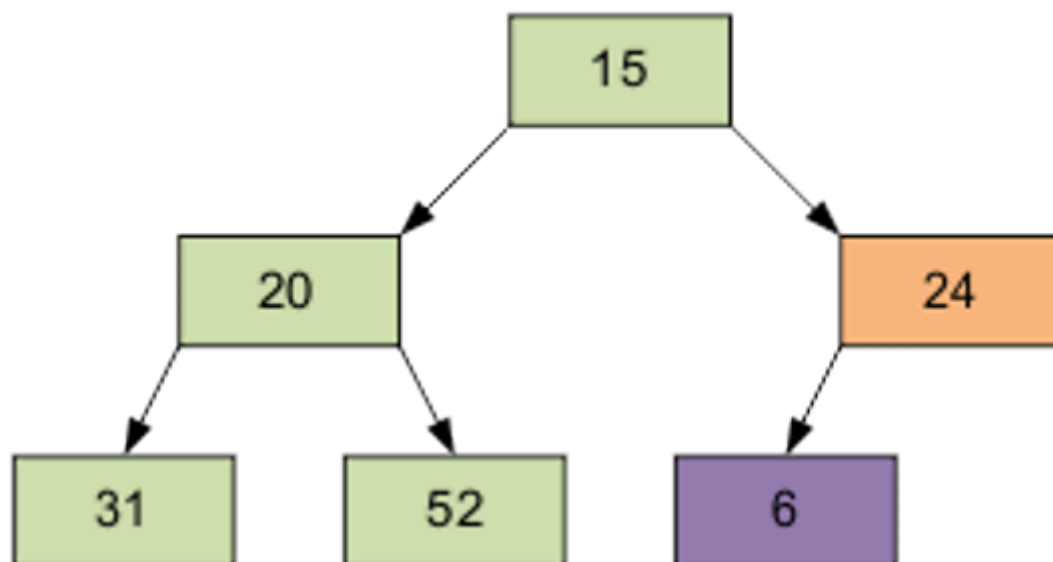
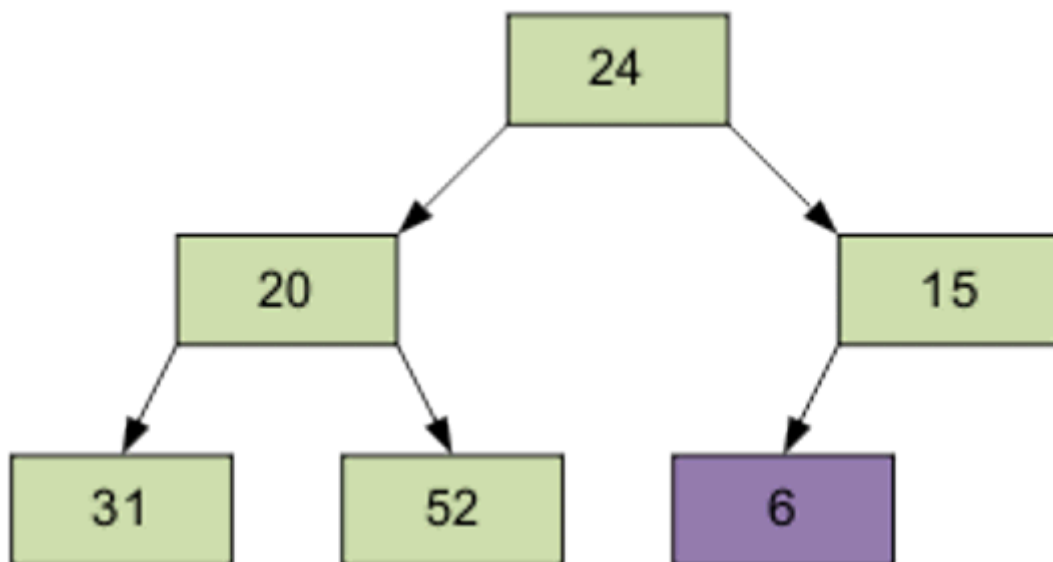


Затем – элемент 0, равный 24.

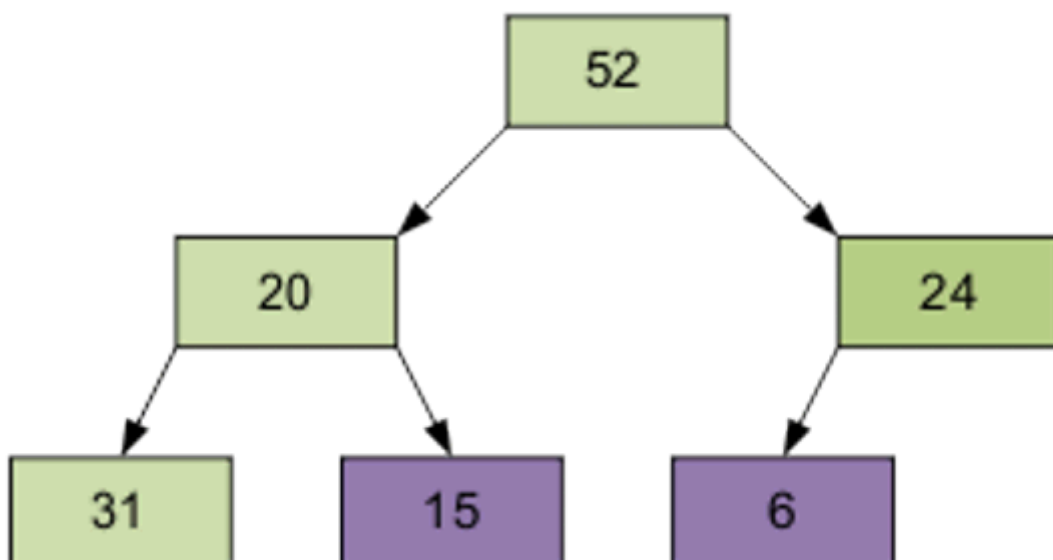


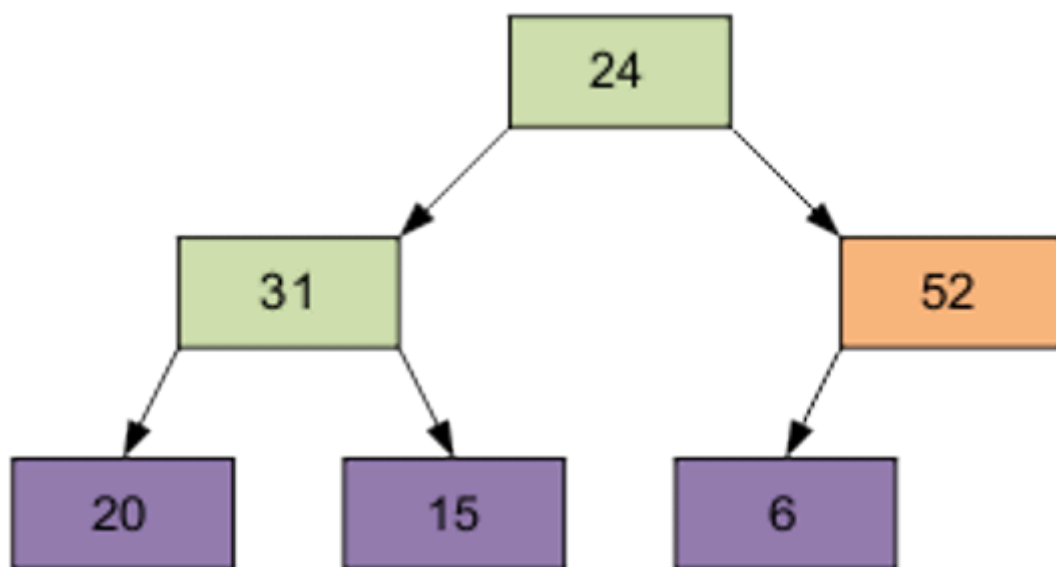
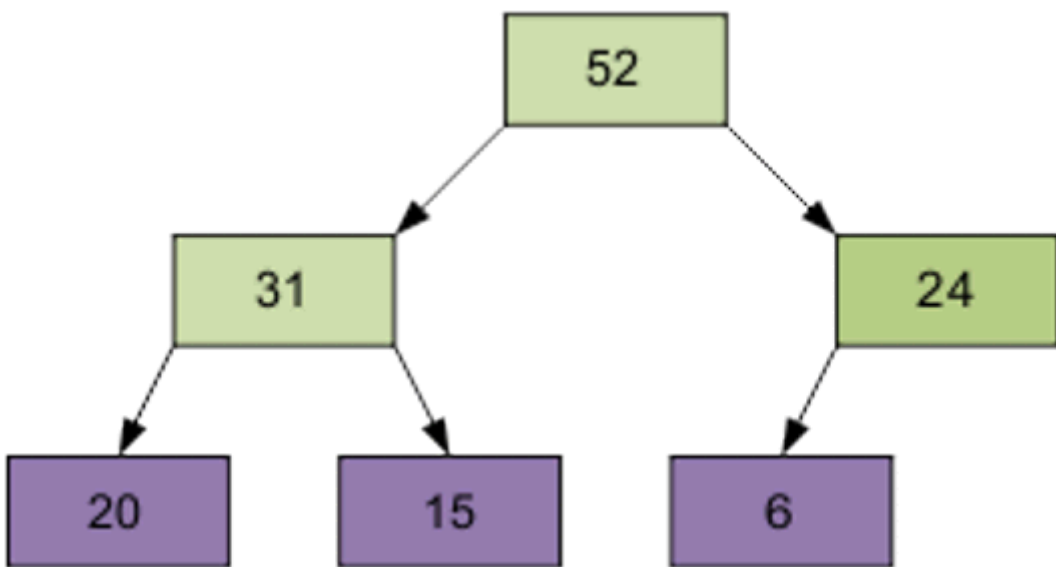
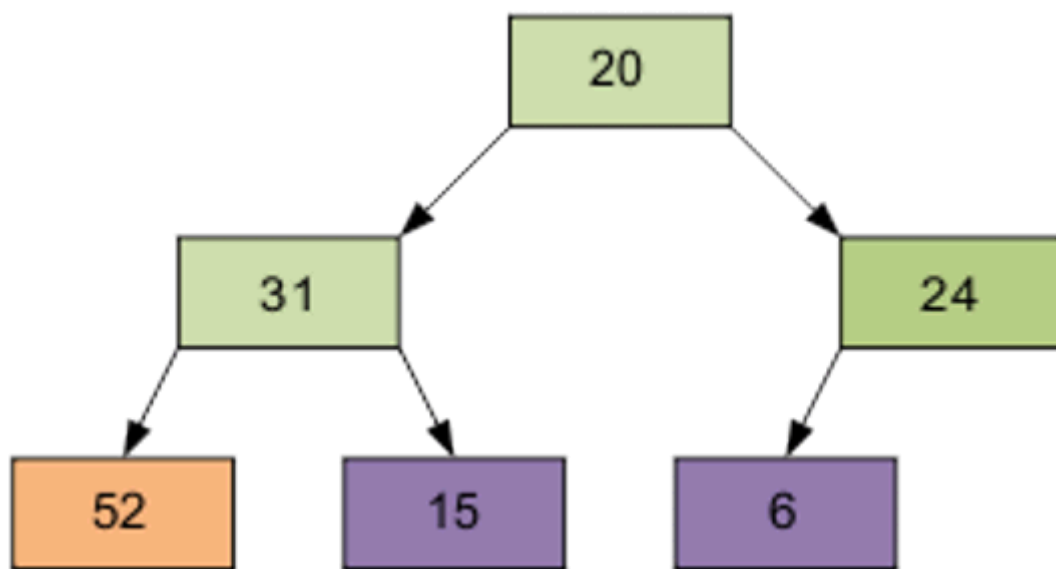
Разумеется, полученный массив еще не упорядочен. Однако процедура просеивания является основой для пирамидальной сортировки. В итоге просеивания наименьший элемент оказывается на вершине пирамиды.

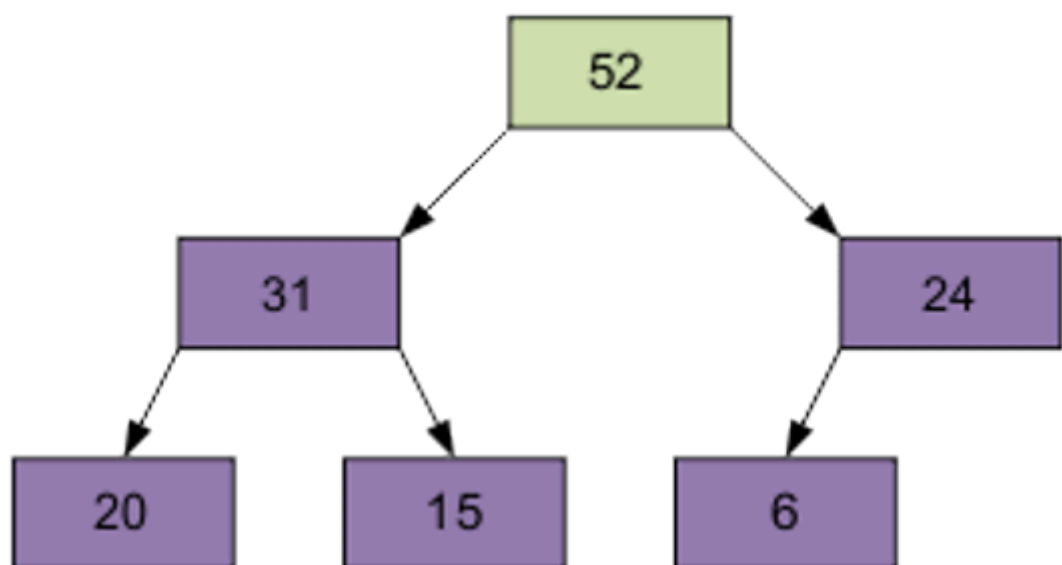
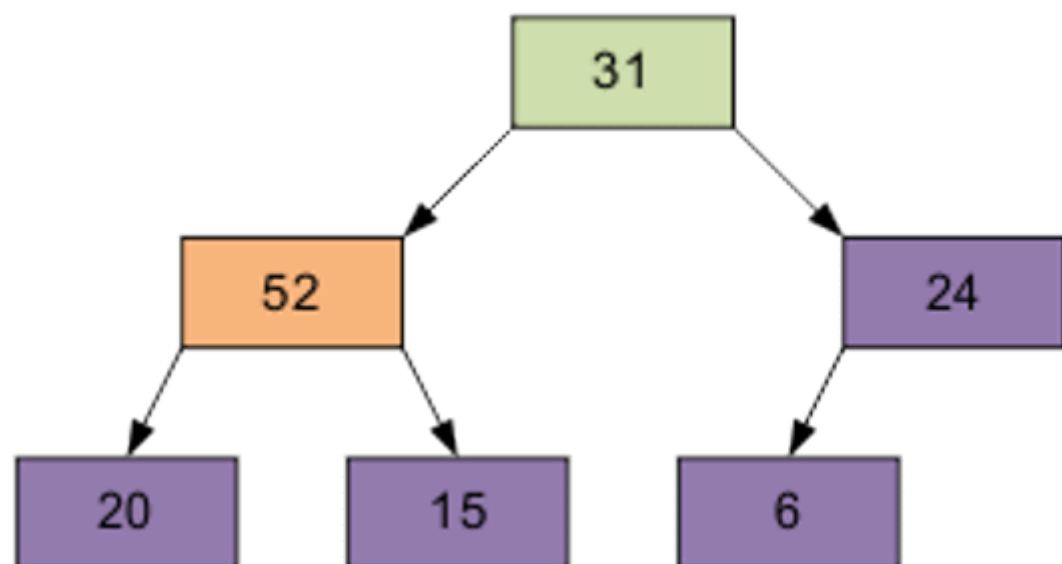
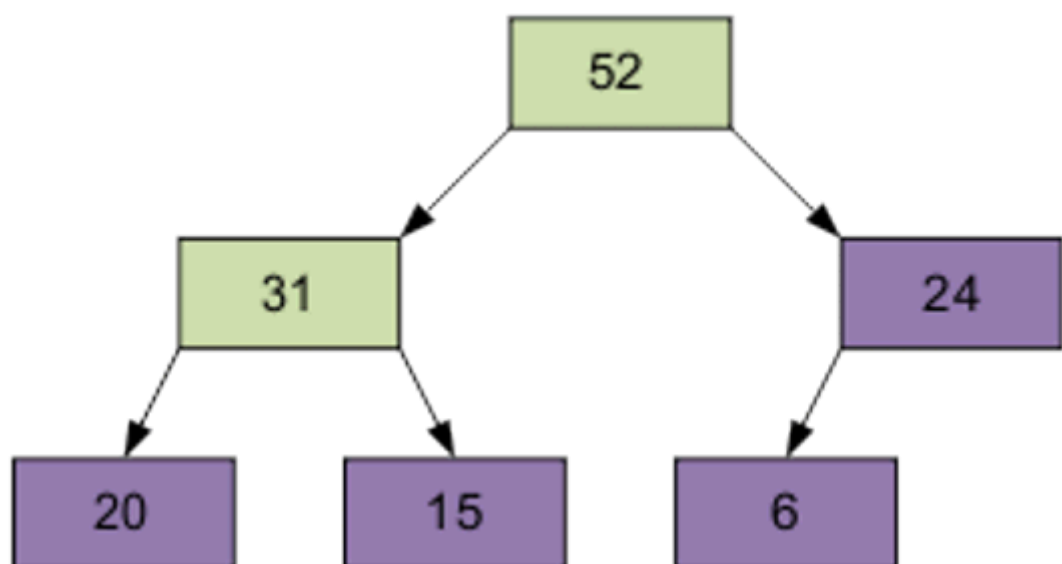
2 этап Сортировка на построенной пирамиде. Берем последний элемент массива в качестве текущего. Меняем верхний (наименьший) элемент массива и текущий местами. Текущий элемент (он теперь верхний) просеиваем сквозь $n-1$ элементную пирамиду. Затем берем предпоследний элемент и т.д.

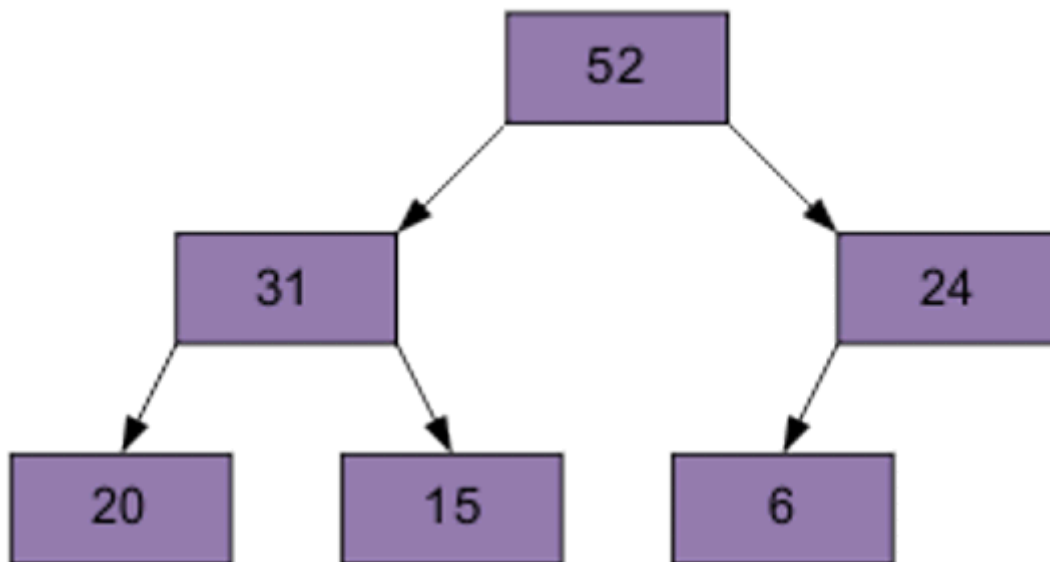


Продолжим процесс. В итоге массив будет отсортирован по убыванию.









Достоинства

- Имеет доказанную оценку худшего случая $O(n \log n)$.
- Сортирует на месте, то есть требует всего $O(1)$ дополнительной памяти (если дерево организовывать так, как показано выше).

Приклад реалізації

Python

```
def heapSort(li):
    def downHeap(li, k, n):
        new_elem = li[k]
        while k <= n/2:
            child = 2*k;
            if child < n and li[child] < li[child+1]:
                child += 1
            if new_elem >= li[child]:
                break
            li[k] = li[child]
            k = child
        li[k] = new_elem

    size = len(li)
    for i in range(round(size/2-1), -1, -1):
        downHeap(li, i, size-1)
    for i in range(size-1, 0, -1):
        temp = li[i]
        li[i] = li[0]
        li[0] = temp
        downHeap(li, 0, i-1)
    return li
```

Литература

- *Левитин А. В.* Глава 6. Метод преобразования: Пирамиды и пирамидальная сортировка // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 275–284. — 576 с. — ISBN 978-5-8459-0987-9

Завдання до лабораторної роботи

Для виконання поставленої мети, необхідно вирішити наступні задачі:

1. Реалізувати алгоритм
2. Протестувати алгоритм для різних вхідних даних.
3. Реалізувати інтерфейс користувача

Контрольні питання

1. Що таке пірамідальне сортування?
2. У чому полягає особливість алгоритму?
3. Аналіз алгоритму пірамідальним сортуванням (найкращий випадок, найгірший, середній).
4. Недоліки алгоритму