Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая автоматизации и робототехники

# Курсовая работа

Дисциплина: Объектно-ориентированное программирование

Тема: Heap sort

Студент гр. 3331506/80401: Краевский Е. И.
Преподаватель: Кузнецова Е. М.
«___»_____2021 г.

Санкт-Петербург
2021

# 1. Введение

Heap sort или пирамидальная сортировка – алгоритм сортировки на основе сравнения. Он был изобретен Дж. Уильямсом (J. W. J. Williams) в 1964 году. Данный алгоритм активно применяется в ядре Linux.

# 2. Описание алгоритма

Пирамидальная сортировка основана на такой структуре данных как двоичная куча. Эта структура данных соблюдает следующие условия:

1. Значение в любой вершине не меньше, чем значения её потомков
2. Глубина всех листьев отличается не более чем на 1 слой.
3. Последний слой заполняется слева направо без «дырок».

Куча может быть представлена двоичным деревом или массивом. На практике используется массив из-за эффективности с точки зрения расхода памяти. В таком случае, если родительский узел хранится в индексе $i$, то левый дочерний элемент может быть вычислен как $2i + 1$, а правый – как $2i + 2$.

Рассмотрим более подробно алгоритм построения двоичной кучи снизу. Сначала следует представить исходный массив в виде кучи, не заботясь о соблюдении ее основного свойства. После чего нужно вызвать *heapify* для всех вершин, у которых есть хотя бы один потомок. Такие вершину будут иметь индекс не больше $n/2$, где $n$ – количество элементов массива. В функции *heapify* вершина сравнивается с потомками, и при необходимости меняется местами. Данная процедура повторяется до тех пор, пока не дойдет до вершины, и массив превратиться в двоичную кучу. Временная оценка такого алгоритма $O(n)$.

Отметим также, что строить кучу можно и сверху вниз. Однако при этом временная оценка будет $O(n \log n)$, т. к. при замене предка и потомка необходимо еще рез проверить основное свойство на уровень выше, и поменять элементы местами при необходимости.

В данном случае куча строиться снизу вверх.

Алгоритм heap sort работает следующим образом:

1. Из входных данных строится двоичная куча.
2. Корень меняется местами с последним элементом кучи и размер кучи уменьшается на 1.
3. Повторение 1 и 2 шага до тех пор, пока размер кучи не станет равен 1.

Код программы представлен в приложении 1.

## 3. Скорость работы алгоритма

Грубую оценку алгоритма можно провести следующим образом: на каждом шаге мы опускаемся по дереву на один уровень, а высота дерева есть $O(\log n)$. Функция $heapify$ вызывается $n$ раз, таким образом сложность алгоритма $O(n \log n)$.

При более точной оценке этап построения кучи занимает время O(n). Для сыновей корневого узла необходимо по одной операции, для внуков уже четыре операции и т. д.

## 4. Исследование времени алгоритма

На рисунке 1 представлен график зависимости времени сортировки от количества элементов в массиве. Массив заполнялся случайными числами.

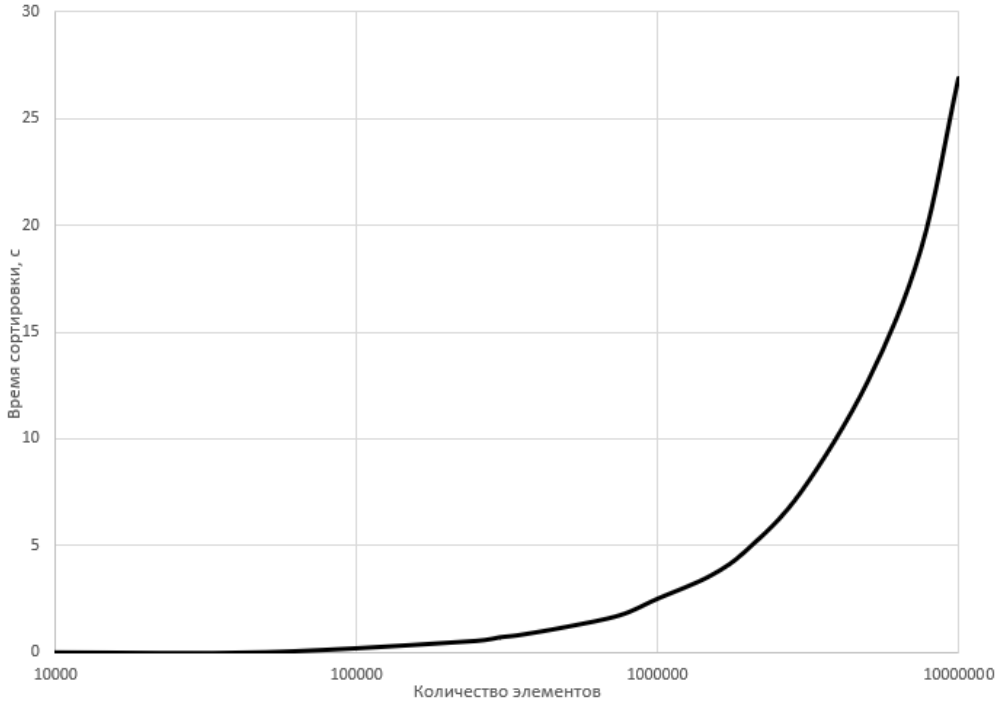


Рисунок 1

## 5. Список литературы

1. Левитин А. Алгоритмы: введение в разработку и анализ – М.: Вильямс, 2006.

2. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ – М.: Вильямс, 2005.

# 6. Приложение

Приложение 1

```
#include <iostream>
using namespace std;

void heapify(int array[], int arr_size, int i)
{
    int largest  = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < arr_size && array[left] > array[largest])
    {
        largest = left;
    }
    if (right < arr_size && array[right] > array[largest])
    {
        largest = right;
    }
    if (largest != i)
    {
        swap(array[i], array[largest]);
        heapify (array, arr_size, largest);
    }
}

void heap_sort(int array[], int arr_size)
{
    for (int i = arr_size / 2 - 1; i >= 0; i--)
    {
        heapify(array, arr_size, i);
    }
    for (int i = arr_size - 1; i >= 0; i--)
    {
        swap(array[0], array[i]);
        heapify(array, i, 0);
    }
}

int main()
{
    return 0;
}
```