

## Лабораторная работа 3. Бинарная куча, фибоначчиева куча.

### Постановка задачи

Реализовать структуру данных «бинарная куча» (*min-heap*) на основе массива. Реализовать функции:

```
struct minheap *minheap_insert(struct minheap *heap,
                               int key, char *value);
struct minheap *minheap_min(struct minheap *heap);
struct minheap *minheap_extractmin(struct minheap *heap);
struct minheap *minheap_decrease_key(struct minheap *heap,
                                      int key, int newkey);
struct minheap *minheap_delete(struct minheap *heap, int key);
```

Реализация функций бинарной кучи соответствует реализации в 9-й лекции 2-го семестра и 4-й лабораторной работе 2-го семестра, «алгоритм Дейкстры» (добавление элемента, извлечение элемента с минимальным приоритетом, уменьшение приоритета ключа). Механизм удаления элемента с любым ключом соответствует показанному для фибоначчиевой кучи.

Реализовать структуру данных «фибоначчиева куча». Реализовать функции:

```
struct fibheap *fibheap_insert(struct fibheap *heap,
                               int key, char *value);
struct fibheap *fibheap_min(struct fibheap *heap);
struct fibheap *fibheap_union(struct fibheap *heap1,
                             struct fibheap *heap2);
struct fibheap *fibheap_extractmin(struct fibheap *heap);
struct fibheap *fibheap_decrease_key(struct fibheap *heap,
                                     struct fibheap *node, int newkey);
struct fibheap *fibheap_delete(struct fibheap *heap, int key);
```

Если лабораторная работа сдается после 23.11, количество баллов за лабораторную снижается на 2, после 30.11. – на 5.

### Экспериментальное исследование

- В качестве ключа использовать целое число типа *int*.
- Для добавления генерировать псевдослучайные числа в диапазоне [10 000, 200 000].
- Заполнение всех таблиц предполагает, что сначала все элементы в указанном количестве будут добавлены в бинарную и фибоначчиеву кучу.
- Для заполнения таблицы 1 необходимо извлечь элемент с минимальным приоритетом сразу после добавления всех элементов в структуру данных (*extractmin*), необходимо уменьшить ключ сразу после добавления всех элементов в структуру данных (*decreasekey*).

- Для заполнения таблицы 2 необходимо извлечь все элементы с минимальным приоритетом из структуры последовательно (*extractmin*) и уменьшить ключ всех элементов в структуре данных (*decreasekey*). Изменять суммарное время работы функция для всех элементов.

Таблица 1. Извлечение элемента с минимальным приоритетом и уменьшение приоритета ключа

#	Кол-во элементов	Время выполнения функции <i>minheap_extractmin</i> , с	Время выполнения функции <i>fibheap_extractmin</i> , с	Время выполнения функции <i>minheap_decreasekey</i> , с	Время выполнения функции <i>fibheap_decreasekey</i> , с
1	50000				
2	100000				
3	150000				
4	200000				

Таблица 2. Извлечение всех элементов из структуры данных путем извлечения элемента с минимальным приоритетом и уменьшение приоритета всех ключей в структуре данных

#	Кол-во элементов	Время выполнения функции <i>minheap_extractmin</i> , с	Время выполнения функции <i>fibheap_extractmin</i> , с	Время выполнения функции <i>minheap_decreasekey</i> , с	Время выполнения функции <i>fibheap_decreasekey</i> , с
1	50000				
2	100000				
3	150000				
4	200000				

## Контрольные вопросы

- Что такое очередь с приоритетом?
- Варианты реализации очереди с приоритетом.
- Структура узла фибоначчиевой кучи.
- Структура фибоначчиевой кучи.
- Максимальная степень узла в фибоначчиевой куче.
- Объяснение механизма работы функций бинарной кучи и фибоначчиевой кучи.
- Операция слияния двух фибоначчиевых куч.
- Вычислительная сложность операций бинарной кучи и фибоначчиевой кучи.
- Анализ вычислительной сложности функций бинарной кучи и фибоначчиевой кучи.
- Алгоритм удаления узла с наименьшим приоритетом (уплотнение списка корней — *consolidate*).
- Распространенность и области практического применения фибоначчиевых куч.