|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Алгоритмы и структуры данных |
| ИНСТИТУТ | Институт перспективных технологий и индустриального программирования |
| КАФЕДРА | Кафедра индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Текущий контроль |
| ПРЕПОДАВАТЕЛЬ | Преснецова Виктория Юрьевна, Яковлев Михаил Сергеевич, Дворецкий Артур Геннадьевич, Гиматдинов Дамир Маратович |
| СЕМЕСТР | 2 семестр, 2024-2025 гг. |

**Рабочая тетрадь 3.**

**Двоичная куча. Очередь с приоритетом. Пирамидальная сортировка**

|  |
| --- |
| **Требования** |
| 1. Код должен быть оптимизирован для производительности и использования ресурсов. 2. Необходимо избегать избыточных вычислений и памяти. 3. Комментарии должны объяснять сложные участки кода и логику работы программы. |

**1. Двоичная куча**

|  |  |
| --- | --- |
| **Теоретический материал** | |
| Куча (англ. heap) - это специализированная структура данных типа дерево, которая удовлетворяет свойству кучи: если B является узлом-потомком узла A, то ключ(A) ≥ ключ(B). Из этого следует, что элемент с наибольшим ключом всегда является корневым узлом кучи, поэтому иногда такие кучи называют max-кучами (в качестве альтернативы, если сравнение перевернуть, то наименьший элемент будет всегда корневым узлом, такие кучи называют min-кучами). Не существует никаких ограничений относительно того, сколько узлов-потомков имеет каждый узел кучи, хотя на практике их число обычно не более двух.  Кучи обычно реализуются в виде массивов, что исключает наличие указателей между её элементами.  Над кучами обычно проводятся следующие операции:   * найти максимум или найти минимум: найти максимальный элемент в max-куче или минимальный элемент в min-куче, соответственно * удалить максимум или удалить минимум: удалить корневой узел в max- или min-куче, соответственно * увеличить ключ или уменьшить ключ: обновить ключ в max- или min-куче, соответственно * добавить: добавление нового ключа в кучу. * слияние: соединение двух куч с целью создания новой кучи, содержащей все элементы обеих исходных.   **Реализация двоичной кучи на основе массива** | |
| **Пример 1** | |
| ***Задача:*** | |
|  | Реализация двоичной минимальной кучи |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
| **Задание 1 (1 балл)** | |
| ***Задача:*** | |
|  | На вход программе подается массив, элементы которого расположены в произвольном порядке. Создать из массива максимальную двоичную кучу (см. для примера слайды 4 -10 лекции). |
| ***Решение:*** | |
|  | Снимок экрана 2025-04-10 в 11.23.11 |
| ***Ответ:*** | |
|  | Снимок экрана 2025-04-10 в 11.23.41 |
| **Задание 2 (1 балл)** | |
| ***Задача:*** | |
|  | Создать класс «Двоичная куча» (максимальная куча). Реализовать для класса: конструктор, формирующий кучу из произвольного массива, конструктор по умолчанию (создание пустой кучи). Для класса реализовать методы: поиск максимума, удаление максимума, добавление нового элемента в кучу. |
| ***Решение:*** | |
|  | Снимок экрана 2025-04-11 в 01.14.55Снимок экрана 2025-04-11 в 01.15.15Снимок экрана 2025-04-11 в 01.15.24 |
| ***Ответ:*** | |
|  | Снимок экрана 2025-04-11 в 01.16.42 |
| **Задание 3\*** | |
| ***Задача:*** | |
|  | Дополните задание 2, добавив для класса метод слияния двух куч. |
| ***Решение:*** | |
|  | Снимок экрана 2025-04-11 в 02.16.15Снимок экрана 2025-04-11 в 02.16.32 |
| ***Ответ:*** | |
|  | Снимок экрана 2025-04-11 в 02.16.40 |
| **Задание 4 (1 балл)** | |
| ***Задача:*** | |
|  | Задача о связывании канатов. Дан список (массив) канатов различной длины, которые требуется связать, по два за раз, в порядке, который приведет к наименьшим затратам. Затраты на соединение двух канатов равны сумме их длин, а общие затраты равны сумме соединения всех канатов. На выходе программа должна выдать порядок связывания канатов и суммарные затраты. Решить задачу на основе использования двоичной кучи (см. слайды 28-29 лекции) |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
| **Задание 5\*** | |
| ***Задача:*** | |
|  | Напишите функцию, которая принимает двоичное дерево в качестве параметра и возвращает *true*, если это двоичная куча (максимальная или минимальная), и *false*, если нет. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |

**2. Очередь с приоритетом**

|  |  |
| --- | --- |
| **Теоретический материал** | |
| **Очередь с приоритетом** - это абстрактный тип данных, описывающий структуру данных, в которой каждый фрагмент обладает определенным приоритетом. В отличие от очереди, освобождающей элементы по принципу «первым вошел, первым вышел», очередь с приоритетом обслуживает элементы в соответствии с приоритетом: сначала она удаляет данные с наивысшим приоритетом, затем данные с последующими по рангу приоритетами (или, наоборот, начинает с наименьших значений).  Очередь с приоритетом поддерживает две обязательные операции — добавить элемент и извлечь максимум (минимум). Предполагается, что для каждого элемента можно вычислить его приоритет — действительное число.  Основные методы, реализуемые очередью с приоритетом, следующие:  **insert(ключ, значение)** — добавляет пару (ключ, значение) в хранилище;  **extract\_minimum()** — возвращает пару (ключ, значение) с минимальным значением ключа, удаляя её из хранилища.  При этом меньшее значение ключа соответствует более высокому приоритету.  В некоторых случаях более естественен рост ключа вместе с приоритетом. Тогда второй метод можно назвать **extract\_maximum()**.  В качестве примера очереди с приоритетом можно рассмотреть список задач работника. Когда он заканчивает одну задачу, он переходит к очередной — самой приоритетной, то есть выполняет операцию извлечения максимума. Начальник добавляет задачи в список, указывая их приоритет, то есть выполняет операцию добавления элемента.  **Одной из популярных реализаций очереди с приоритетом является куча.**  Структура узла для очереди с приоритетом, реализованной на базе связного списка  struct Node {  int data; // Значение элемента  int priority; // Приоритет элемента (меньшее значение - более высокий приоритет)  Node\* next; // Указатель на следующий узел  }; | |
| **Задание 6 (1 балл)** | |
| ***Задача:*** | |
|  | Реализовать очередь с приоритетом без использования кучи, используя связный список. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
| **Задание 7\*** | |
| ***Задача:*** | |
|  | Реализовать очередь с приоритетом на базе двоичной кучи. На основе очереди с приоритетом сделать todo list[[1]](#footnote-0), для которого пользователь либо добавляет новое задание, имеющее название (одно слово) и приоритет (целое число),  либо запрашивает задание с максимальным приоритетом из списка дел и делает его (помечает как выполненное, выводит за пределы очереди). Предусмотреть возможность редактирования задания. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |

**3. Пирамидальная сортировка**

|  |
| --- |
| **Теоретический материал** |
| Пирамидальная сортировка (Heap Sort) – это эффективный алгоритм сортировки, основанный на двоичной куче (Binary Heap). Он работает по принципу построения кучи и поочередного извлечения максимального (или минимального) элемента.    Основные принципы работы Heap Sort  1. Преобразование массива в двоичную кучу (Heapify)  Если сортировка идет по возрастанию, то строится Max-Heap (где каждый родитель больше потомков).  Если сортировка идет по убыванию, то строится Min-Heap (где каждый родитель меньше потомков).  2. Извлечение максимального элемента (он всегда находится в корне Max-Heap).  Меняем корневой элемент с последним в куче.  Уменьшаем размер кучи и восстанавливаем свойства кучи (heapify).  3. Повторяем процесс до тех пор, пока не останется один элемент. |

|  |  |
| --- | --- |
| **Задание 8 (2 балла)** | |
| ***Задача:*** | |
|  | Реализовать алгоритм пирамидальной сортировки массива (слайды 19-25 лекции) |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |

1. «ToDo list» — приложение, которое позволяет пользователям добавлять, редактировать и удалять задачи, над которыми они хотят работать, а также отмечать задачи как выполненные, не удаляя их. [↑](#footnote-ref-0)