|  |  |
| --- | --- |
| 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 4.**

**Префиксное дерево. Дерево Хаффмана. Примеры на жадные алгоритмы**

|  |
| --- |
| **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. Префиксное дерево**

|  |
| --- |
| **Теоретический материал** |
| Префиксное дерево (бор, trie) - это структура данных, используемая для хранения динамических множеств или ассоциативных массивов, где ключами являются строки. Оно представляет собой корневое дерево, в котором каждая вершина соответствует символу строки.  Некоторые узлы префиксного дерева выделены (на рисунке они подписаны цифрами) и считается, что префиксное дерево содержит данную строку-ключ [тогда и только тогда](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D0%B3%D0%B4%D0%B0_%D0%B8_%D1%82%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE_%D1%82%D0%BE%D0%B3%D0%B4%D0%B0" \o "Тогда и только тогда), когда эту строку можно прочитать на пути из корня до некоторого (единственного для этой строки) выделенного узла. В некоторых приложениях удобно считать все узлы дерева выделенными.    Основные свойства:   1. Иерархическая структура - строки хранятся в виде пути от корня к листу. 2. Общий префикс - если несколько строк имеют общий префикс, они разделяют начальную часть пути. 3. Операции поиска, вставки и удаления выполняются за O(L), где L - длина строки.   Основные операции:   1. Вставка (insert): добавление слова в дерево. 2. Поиск (search): проверка наличия слова. 3. Проверка префикса (startsWith): проверка существования строк, начинающихся с данного префикса. |

|  |  |
| --- | --- |
| **Пример 1** | |
| ***Задача:*** | |
|  | Реализуйте поиск в префиксном дереве |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
| **Задание 1 (1.5 балла)** | |
| ***Задача:*** | |
|  | Реализуйте функцию, которая находит количество слов в префиксном дереве, начинающихся с заданного префикса. |
| ***Решение:*** | |
|  | Снимок экрана 2025-04-30 в 14.19.51Снимок экрана 2025-04-30 в 14.19.51Снимок экрана 2025-04-30 в 14.20.02 |
| ***Ответ:*** | |
|  | Снимок экрана 2025-04-30 в 14.27.26 |
| **Задание 2 (1.5 балла)** | |
| ***Задача:*** | |
|  | Реализуйте операцию удаления слова из trie и проверьте, осталось ли оно после удаления. |
| ***Решение:*** | |
|  | Снимок экрана 2025-05-01 в 19.37.13Снимок экрана 2025-05-01 в 19.37.28 |
| ***Ответ:*** | |
|  | Снимок экрана 2025-05-01 в 19.37.41 |
| **Задание 3\*** | |
| ***Задача:*** | |
|  | Вам дан список слов. Найдите слово, у которого наибольшее количество других слов списка являются его префиксами.  *Вход:*  a  ab  abc  abcd  abcdef  bcd  *Выход:*  abcdef (Префиксы: a, ab, abc, abcd) |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |

**2. Жадные алгоритмы. Дерево Хаффмана**

|  |  |
| --- | --- |
| **Теоретический материал** | |
| Жадные алгоритмы - это алгоритмические методы, которые принимают локально оптимальные решения на каждом шаге с надеждой, что они приведут к глобально оптимальному решению.  Основной принцип жадных алгоритмов заключается в построении решения поэтапно, где на каждом шаге выбирается лучший возможный вариант. При этом выбор текущего шага никак не изменяется в дальнейшем, что отличает жадные алгоритмы от других подходов, таких как динамическое программирование или метод перебора с откатом. Такой подход оказывается эффективным, если задача обладает свойствами жадного выбора и оптимальной подструктуры. Жадный выбор означает, что выбор локально наилучшего решения на каждом шаге приведёт к оптимальному глобальному решению. Оптимальная подструктура подразумевает, что любое оптимальное решение задачи может быть получено из оптимальных решений её подзадач.  Применение жадных алгоритмов обосновано в тех случаях, когда можно доказать, что жадный выбор всегда приводит к глобально оптимальному решению. Однако, если такой гарантии нет, жадные алгоритмы могут давать лишь приближённые результаты. Это отличает их от динамического программирования, которое строит решение путём комбинирования ранее вычисленных оптимальных значений. Динамическое программирование требует значительных вычислительных ресурсов, тогда как жадные алгоритмы, как правило, имеют меньшую временную сложность и требуют меньше памяти.  Жадные алгоритмы применяются в самых разных областях, включая теорию графов, комбинаторную оптимизацию и задачи планирования. Например, в алгоритме Дейкстры для нахождения кратчайшего пути жадный выбор заключается в том, чтобы на каждом шаге выбирать вершину с минимальным расстоянием от начальной вершины. В алгоритмах построения минимального остовного дерева, таких как алгоритмы Крускала и Прима, жадный подход заключается в том, чтобы последовательно добавлять рёбра с минимальным весом, не создавая циклов.  *Алгоритм Хаффмана*  Обычно для хранения данных и передачи сообщений используются коды фиксированной длины, например, код ASCII.  Множество символов представляются некоторым количеством кодовых слов равной длины, которая для кода ASCII равна восьми битам (1 байт). При этом для всех сообщений с одинаковым количеством символов требуется одинаковое количество битов при хранении и одинаковая ширина полосы пропускания при передаче.  Конечно, если сообщение написано, скажем, на английском языке, то вероятность появления в нем букв "z" намного меньше, чем вероятность появления букв "e". Это означает, что если для представления буквы "e" использовать более короткое кодовое слово, чем для представления буквы "z", то можно ожидать, что в среднем для хранения сообщения потребуется меньше памяти, а для его передачи - меньшая ширина канала.  В коде ASCII сообщение "easily" кодируется следующим образом:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 01100101 | 01100001 | 01110011 | 01101001 | 01101100 | 01111001 | | e | a | s | i | l | y |   для чего требуется 48 бит, в то время как пpи использовании кода со следующим представлением символов   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 1001 | 0 | 1010 | 11001 | 11010 | 1011 | | a | e | i | l | s | y |   то же самое сообщение можно закодировать следующим образом    используя только 23 бита.  Кодовые слова должны быть выбраны так, чтобы никакое из них не было префиксом любого другого кодового слова. Благодаря этому условию гарантируется возможность однозначного декодирования определённого закодированного текста.  В классической статье, опубликованной в 1952 г. Дэвид Хаффман описал алгоритм поиска множества кодов, которые минимизируют ожидаемую длину сообщений пpи условии, что известны вероятности появления каждого символа.  Существенно, что в этом методе пpи определении длины кодовых слов символам, имеющим меньшую вероятность появления, ставятся в соответствие более длинные кодовые слова. После этого остается образовать некоторый однозначно декодируемый код с кодовыми словами надлежащей длины.  Хаффман в заключительном разделе работы отождествляет однозначное множество кодовых слов с двоичным деревом. Каждый лист дерева соответствует одному из символов. Глубина этого листа, т.е. его расстояние от корня, - это длина кодового слова соответствующего символа.    Цифры кодового слова являются адресом этого листа, т.е. последовательностью инструкций для продвижения от корня к листу, например, команда "0"- - двигаться влево, а "1" - двигаться вправо. Тогда каждой вершине дерева будет приписано двоичное слово, описывающее, как добраться к этой вершине от корня. Самому корню соответствует пустое слово "0".  Хаффман пишет: "Так как объединение сообщений в составные сообщения подобно слиянию струек, ручейков и речушек в одну большую реку, описанную выше процедуру можно рассматривать по аналогии с расстановкой знаков жуком-плавунцом в каждом месте впадения притоков по пути его перемещения вниз по течению ... искомым будет код, который должен помнить плавунец, чтобы совершить обратный путь против течения".  **Алгоритм Хаффмана:**  **1 этап:**  Множество символов располагается в порядке уменьшения вероятностей их появления. Каждому из символов будет соответствовать лист дерева, следовательно, можно представить себе этот этап процесса как построение линейного списка, содержащего листья будущего дерева.  **2 этап:**  Производится повторяющееся сокращение числа максимальных непересекающихся поддеревьев посредством объединения двух "легчайших" деревьев для получения нового составного дерева.  Листья любого бинарного дерева образуют префиксный код, и, наоборот, для всякого префиксного кода существует такое дерево, что слова кода соответствуют его листьям. | |
| **Задание 4 (2 балла)** | |
| ***Задача:*** | |
|  | Компания разрабатывает систему сжатия данных для передачи текстовой информации. Вам необходимо реализовать алгоритм Хаффмана, чтобы минимизировать количество бит, используемых для представления текста.  Реализовать следующие методы:  а) Метод построения дерева;  б) Метод построения списка частот символов;  в) Метод шифрования (сжатия);  г) Метод дешифровки;  д) подсчет коэффициента сжатия.  Пример работы программы представлен ниже |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
| **Задание 5\*** | |
| ***Задача:*** | |
|  | Вы работаете в компании, разрабатывающей систему оптимизированной передачи текстовой информации по каналам связи с ограниченной пропускной способностью. Ваша задача -не просто закодировать текст с помощью алгоритма Хаффмана, но и минимизировать размер передаваемых данных, включая сам словарь кодировки.  Для этого требуется:  1. Закодировать строку S с помощью алгоритма Хаффмана.  2. Минимизировать размер передаваемого сообщения, учитывая необходимость передачи словаря кодировки.  3. Оценить, можно ли сэкономить место за счёт битового хранения словаря. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |

**3. Примеры на жадные алгоритмы. Задача о размене монет**

|  |
| --- |
| **Теоретический материал** |
| Некоторые классические задачи, решаемые жадными алгоритмами:  1. Задача о размене монет - выбор минимального количества монет для получения нужной суммы.  2. Задача о рюкзаке (фракционная версия) - выбор предметов с максимальным удельным весом (ценность/вес).  3. Задача о покрытии отрезков - выбор минимального количества интервалов для покрытия множества точек.  4. Алгоритмы минимального остовного дерева:  Алгоритм Крускала - добавляет рёбра в порядке возрастания веса.  Алгоритм Прима - строит остовное дерево, начиная с одной вершины.  5. Алгоритм Дейкстры - поиск кратчайшего пути в графе с неотрицательными весами рёбер.  **Задача о размене монет**  **Условие:**  Дано некоторое количество монет различных номиналов и сумма, которую необходимо разменять. Нужно найти минимальное количество монет, необходимое для размена данной суммы, используя жадный алгоритм.  **Жадный подход:**  1. Отсортируем монеты по убыванию номинала.  2. Начнём размен с монеты наибольшего номинала.  3. На каждом шаге выбираем максимальное количество монет данного номинала, не превышая сумму.  4. Переходим к монете меньшего номинала и повторяем процесс, пока сумма не будет разменяна.  **Ограничение:**  Жадный алгоритм работает корректно, если система монет обладает свойством жадного выбора, т.е. любой размен можно представить как сумму наибольших возможных номиналов (например, для стандартных монет 1, 5, 10, 25, 50 он работает, но для произвольных наборов может давать неоптимальный результат). |

|  |  |
| --- | --- |
| **Пример 2** | |
| ***Задача:*** | |
|  | Реализовать задачу о размене монет |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
| **Задание 6 (2 балла)** | |
| ***Задача:*** | |
|  | Дано N задач, каждая из которых имеет время начала и окончания. Нужно выбрать максимальное количество непересекающихся задач, чтобы выполнить их в одном рабочем дне. Сортируйте задачи по времени окончания и выбирайте те, которые начинаются после завершения предыдущей. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |
| **Задание 7\*** | |
| ***Задача:*** | |
|  | Есть N заказов, которые нужно доставить, и K курьеров. Каждый курьер может взять несколько заказов, но его общий маршрут не должен превышать T километров. Требуется минимизировать количество задействованных курьеров. Сортируйте заказы по дальности и жадно распределяйте их между курьерами. |
| ***Решение:*** | |
|  |  |
| ***Ответ:*** | |
|  |  |