Алгоритмы и структуры данных. Обучение в записи

Задание 1. Сортировка массива задач по приоритету (Heap Sort)
Вам дан список задач, каждая из которых имеет приоритет (целое число). Напишите функцию `sortTasksByPriority`, которая сортирует задачи по убыванию приоритета с использованием сортировки кучей (Heap Sort).

```
import java.util.Arrays;


public class HeapSortTasks {
    public static void heapify(int[] arr, int n, int i) {
        // ваша реализация
    }

    public static int[] sortTasksByPriority(int[] tasks) {
        // ваша реализация
    }

    public static void main(String[] args) {
        int[] tasks = {3, 1, 4, 2, 5};

System.out.println(Arrays.toString(sortTasksByPriority(tasks))
);
    }
}
```

Подсказка № 1

Сортировка кучей основана на использовании двоичной кучи. Для начала вам нужно построить максимальную кучу из массива задач. Каждый элемент массива будет частью этой кучи.

Подсказка № 2

Функция `heapify` отвечает за поддержание структуры кучи. Она должна находить наибольший элемент среди родителя и его потомков (левого и правого), и если какой-то потомок больше родителя, нужно поменять их местами.

Подсказка № 3

Чтобы пройти по всем элементам массива и построить кучу, начните с родительского элемента, находящегося на середине массива, и спускайтесь вниз. Родительские элементы можно найти по формуле `i = n / 2 - 1`, где `n` — длина массива.

Подсказка № 4

После того как вы построили кучу, вам нужно перемещать наибольший элемент (он находится в корне кучи) в конец массива, а затем снова вызывать функцию `heapify`, чтобы восстановить структуру кучи для оставшихся элементов.

Подсказка № 5

Когда вы перемещаете элемент в конец массива, помните, что оставшаяся часть массива должна снова стать кучей. Поэтому необходимо уменьшить длину массива на 1 и повторить процесс до тех пор, пока все элементы не будут отсортированы.

Эталонное решение:

```
import java.util.Arrays;


public class HeapSortTasks {

   public static void heapify(int[] arr, int n, int i) {

       int largest = i;

       int left = 2 * i + 1;

       int right = 2 * i + 2;


       if (left < n && arr[left] > arr[largest])
```

```
            largest = left;


        if (right < n && arr[right] > arr[largest])
            largest = right;


        if (largest != i) {
            int swap = arr[i];
            arr[i] = arr[largest];
            arr[largest] = swap;
            heapify(arr, n, largest);
        }
    }


    public static int[] sortTasksByPriority(int[] tasks) {
        int n = tasks.length;
        for (int i = n / 2 - 1; i >= 0; i--)
            heapify(tasks, n, i);


        for (int i = n - 1; i >= 0; i--) {
            int temp = tasks[0];
            tasks[0] = tasks[i];
            tasks[i] = temp;
            heapify(tasks, i, 0);
        }


        return tasks;
    }
```

```
    public static void main(String[] args) {
        int[] tasks = {3, 1, 4, 2, 5};
System.out.println(Arrays.toString(sortTasksByPriority(tasks)));
    }
}
```

Задача 2. Сортировка номеров телефонов по разрядам (Radix Sort)

Напишите функцию `sortPhoneNumbers`, которая сортирует номера телефонов в порядке возрастания с использованием поразрядной сортировки (Radix Sort).

```
import java.util.Arrays;

public class RadixSortPhoneNumbers {
    public static void countingSort(long[] arr, int exp) {
        // ваша реализация
    }

    public static long[] sortPhoneNumbers(long[] arr) {
        // ваша реализация
    }

    public static void main(String[] args) {
        long[] phoneNumbers = {9876543210L, 1234567890L,
5555555555L, 1000000000L};

System.out.println(Arrays.toString(sortPhoneNumbers(phoneNumbe
rs)));
```

```
    }

}
```

Подсказка № 1

В поразрядной сортировке (Radix Sort) сортируются числа по каждому разряду, начиная с младшего (единицы) и заканчивая старшим. Для этого потребуется несколько итераций, на каждой из которых числа сортируются по определённому разряду

Подсказка № 2

Для сортировки на каждом разряде используйте сортировку подсчётом (Counting Sort). Вам нужно отсортировать числа по их текущему разряду (например, сначала по единицам, затем по десяткам и так далее). Чтобы выделить разряд, используйте операцию деления и взятия остатка.

Подсказка № 3

Начните с младшего разряда (единицы). Для этого разделите число на 1 (единицы) и возьмите остаток от деления на 10, чтобы получить цифру в этом разряде. После этого сортируйте числа по этой цифре, используя Counting Sort.

Подсказка № 4

После того как отсортируете числа по младшему разряду, переходите к следующему разряду (десятки), деля числа на 10 и снова применяя Counting Sort. Повторяйте этот процесс для всех разрядов, пока не отсортируете по старшему разряду.

Подсказка № 5

Чтобы определить, когда завершить сортировку, найдите максимальное число в списке и продолжайте делить на 10, пока частное не станет меньше 1 (то есть больше не будет разрядов для сортировки).

Эталонное решение:

```
import java.util.Arrays;


public class RadixSortPhoneNumbers {

   public static void countingSort(long[] arr, int exp) {

        int n = arr.length;
```

```
        long[] output = new long[n];
        int[] count = new int[10];

        for (int i = 0; i < n; i++)
            count[(int) ((arr[i] / exp) % 10)]++;

        for (int i = 1; i < 10; i++)
            count[i] += count[i - 1];

        for (int i = n - 1; i >= 0; i--) {
            output[count[(int) ((arr[i] / exp) % 10)] - 1] = arr[i];
            count[(int) ((arr[i] / exp) % 10)]--;
        }

        for (int i = 0; i < n; i++)
            arr[i] = output[i];
    }

    public static long[] sortPhoneNumbers(long[] arr) {
        long max = Arrays.stream(arr).max().getAsLong();

        for (int exp = 1; max / exp > 0; exp *= 10)
            countingSort(arr, exp);

        return arr;
    }
```

```
    public static void main(String[] args) {

        long[] phoneNumbers = {9876543210L, 1234567890L, 5555555555L,
1000000000L};


System.out.println(Arrays.toString(sortPhoneNumbers(phoneNumbers)));

    }

}
```

Задача 3. Подсчёт количества букв в строке (Counting Sort)

Напишите функцию `countLetters`, которая подсчитывает количество каждой буквы в строке и выводит их по порядку алфавита. Функция должна игнорировать регистр букв.

```
import java.util.Map;

import java.util.TreeMap;



public class CountingSortLetters {

    public static void countLetters(String text) {

        // ваша реализация



    }



    public static void main(String[] args) {

        String text = "Hello World";

        countLetters(text);

    }

}
```

Подсказка № 1

Прежде чем подсчитывать количество букв, преобразуйте всю строку в нижний регистр. Это поможет игнорировать различие между заглавными и строчными буквами. Используйте метод `toLowerCase()` для преобразования строки.

Подсказка № 2

Для подсчёта количества каждой буквы создайте массив длиной 26, где каждая позиция соответствует одной букве английского алфавита. Например, индекс 0 будет соответствовать букве `'a'`, индекс 1 — букве `'b'` и так далее.

Подсказка № 3

Проходите по каждому символу строки. Если символ является буквой, найдите его индекс в массиве с помощью выражения `c - 'a'` (где `c` — это текущий символ). Увеличьте значение в соответствующем индексе на 1.

Подсказка № 4

После того как вы посчитаете количество каждой буквы, выведите только те буквы, которые встречаются хотя бы один раз. Пройдитесь по массиву с индексами и выводите буквы с их количеством, используя преобразование индекса обратно в символ (например, `(char) (i + 'a')`).

Подсказка № 5

Чтобы сохранить порядок вывода букв по алфавиту, используйте структуру данных, такую как `TreeMap`. Она автоматически отсортирует буквы по алфавиту, что упрощает вывод результата.

Эталонное решение:

```
import java.util.Map;

import java.util.TreeMap;



public class CountingSortLetters {

   public static void countLetters(String text) {

       text = text.toLowerCase();

       int[] count = new int[26];
```

```
        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                count[c - 'a']++;
            }
        }

        Map<Character, Integer> letterCount = new TreeMap<>();
        for (int i = 0; i < 26; i++) {
            if (count[i] > 0) {
                letterCount.put((char) (i + 'a'), count[i]);
            }
        }

        for (Map.Entry<Character, Integer> entry :
letterCount.entrySet()) {
            System.out.println(entry.getKey() + ": " +
entry.getValue());
        }
    }

    public static void main(String[] args) {
        String text = "Hello World";
        countLetters(text);
    }
}
```