Задания для курса "Современные управляемые языки программирования"

Иванов А.Б.

Осенний семестр 2023-2024

О заданиях

Предлагается 7 заданий. Итоговая оценка равна количеству сделанных заданий.

Задания должны быть сделаны на языках C# или Java. При желании можно использовать Kotlin на JVM. Язык выбираете сами. Для написания программ рекомендуется использовать следующие среды рзработки (IDE):

- Microsoft Visual Studio (на Windows для C#);
- Rider (на Linux/MacOS для C#);
- Intellij IDEA (для Java или Kotlin).

При создании нового проекта Java в Intellij IDEA рекомендуется использовать систему сборки Maven.

Настоятельная просьба сделать задания в Git проекте на GitHub или GitFlic. Сделайте приватный проект и поделитесь с с пользователем @einsamhauer.

Очень рекомендуется сделать к своим заданиям тесты. Как это делается можно посмотреть в примерах:

- https://github.com/mmalgo/Stack для C#
- https://github.com/mmalgo/IntStack для Java

В обоих проектах для примера реализован минимальный стек с небольшим количеством тестов. (Для Java это стек для int.)

1 Бинарный поиск

Реализуйте обобщенный метод бинарного поиска в массиве используя Comparator/Comparer. Реализация может выглядеть так:

```
public class BinarySearch
{
    public static T Find<T>(T[] array, IComparer<T> comparer, T element)
    {
        ...
    }
}
```

для C#. Или так для Java:

```
public class BinarySearch {
    public static <T> T find(T[] array, Comparator<T> comparator, T element) {
        ...
    }
}
```

2 QuickSort

Реализуйте QuickSort с использованием Comparator/Comparer. Реализация должна использовать хотя бы часть из следующих идей:

1. 3-частное разбиение
2. один рекурсивный вызов вместо двух
3. переключение на сортировку вставками на массивах небольшого размера
4. выбор опорного элемента как медиана из трех

Реализация может выглядеть так:

```
public class QuickSort
{
    public static void Sort<T>(T[] array) where T : IComparable<T> { ... }
    public static void Sort<T>(T[] array, IComparer<T> comparer) { ... }
}
```

для C#. Или так для Java:

```
public class QuickSort {
    public static <T extends Comparable<T>> void sort(T[] array) { ... }
    public static <T> void sort(T[] array, Comparator<T> comparator) { ... }
}
```

Реулизуйте Comparer/Comparator, который подсчитывает количество сравнений. Сравните количество сравнений с реализацией сортировки из стандратной библиотеки.

3 Калькулятор

Реализуйте алгоритм разбора и вычисления арифметического выражения ("калькулятор"). Поддерживаем float числовые значения, бинарные операции $+, -, *, /$ и скобки.

Реализацию можно разбить на три шага:

1. Произвести разбор строки на токены (число, оператор, скобка)
2. С помощью алгоритма сортировочной станции получить выражение в обратной польской записи
3. Реализовать стековый калькулятор

4 Игра в 15

Реализуйте решатель для игры в 15 (https://ru.wikipedia.org/wiki/Игра_в_15) для досок 3x3, 4x4, 5x5, ...

Программа генерирует случайную позицию (допустимую). После этого находит последовательность ходов, приводящую в стартовую позицию.

Способ решения - алгоритм A^*, по сути - нахождение пути в графе, где вершины - позиции, а ребра - ходы из одной позиции в другую. Граф очень большой и хранить его не надо. Для поиска используем очередь с приоритетами, где приорит это

$$a * manhattan + b * moves,$$

где $manhattan$ - манхеттеновское расстояние от текущей позиции до стартовой, а $moves$ - количество уже сделанных ходов. Проанализировать (поэксперементировать) поведение программы в зависимости от a и b.

5 Результаты торгов

Скачайте файл http://ti.math.msu.su/wiki/lib/exe/fetch.php?media=managed:trades.zip.

Файл содержит примерно 1.67 миллиона сделок за 1 торговый день Московской биржи, отсортированых по времени совершения.

Формат файла: 1 строка на каждую сделку, поля разделены символом табуляции "\t":

- TRADENO - номер сделки
- TRADETIME - время
- SECBOARD - площадка
- SECCODE - код ценной бумаги
- PRICE - цена

- VOLUME - количество бумаг в сделке
- ACCRUEDINT - накопленный купонный доход для облигаций
- YIELD - доходность для облигаций
- VALUE - сумма сделки

Напишите программу, которая прочитает этот файл и вычислит для бумаг, торгуемых на площадках TQBR и FQBR, цену открытия и закрытия (цена первой и последней сделки соответственно). Выведите 10 бумаг с максимальным ростом за день и 10 бумаг с максимальным падением. Для каждой из них выведите изменение цены в процентах, количество и общий объем сделок в рублях.

Для решения используйте LINQ в C# и Stream API в Java. Методы, которые могут понадобиться:

```
// C#
.Where, .Select, .SortBy, .GroupBy,
.Count, .Sum, .Min, .Max, .MinBy, .MaxBy, .Aggregate,
.Take, .Skip, .TakeLast, .ToList

// Java
. filter , .map, .reduce, .sorted, . limit , .skip, . toList , . collect
Collectors: . filtering , .mapping, .reducing, .maxBy, .minBy, .counting, .toList
```

Читать файл можно следующим образом:

```
// C#
using (var stream = new FileStream("filename", FileMode.Open, FileAccess.Read))
using (var reader = new StreamReader(stream))
{
    string line = reader.ReadLine();
    var parts = line.Split('\t');
    ...
}
```

```
// Java
try (
    var reader = new FileReader("file.txt");
    var bufferedReader = new BufferedReader(reader))

    String line = bufferedReader.readLine();
    var parts = line.split("\t");
    ...
} catch (IOException e) {
    throw new RuntimeException(e);
}
```

6 Целочисленное сжатие

Реализуйте сжатие (и распаковку) массива целых чисел (int[]) с помощью алгоритма VByte:

```
public int [] encode(int [] array);
public int [] decode(int [] array);
```

. (Закодированный массив это просто набор байтов, но это набор байтов можно интерпретировать как int[].)

Кодировка устроена следующим образом:

- Числа в диапазоне $[0, 2^7)$ кодируются одним байтом. 7 младших битов используются для кодирования. Старший бит - 1.
- Числа в диапазоне $[2^7, 2^{14})$ кодируются двумя байтами. В каждом байте используется 7 младших битов. В первом байте старший бит - 0, во втором - 1.
- Числа в диапазоне $[2^{14}, 2^{21})$ кодируются тремя байтами. В каждом байте используется 7 младших битов. В первых двух байтах старший бит - 0, во втором - 1.
- Числа в диапазоне $[2^{14}, 2^{21})$ кодируются тремя байтами. В каждом байте используется 7 младших битов. В первых двух байтах старший бит - 0, во третьем - 1.
- и так далее.

Для реализации вам понадобяться битовые операции.

7 Разреженные матрицы

Реализуйте класс разреженной матрицы с помощью сжатого представления CSR (https://ru.wikipedia.org/wiki/Разреженная_матрица). Реализуйте операцию умножения таких матриц. Чтобы немного упростить задачу можно ограничится квадратными матрицами.