

Тестовое задание, Цифровой Светофор

1. Введение

Цифровой светофор - светофор, оснащенный индикатором цвета и циферблатом. Индикатор цвета имеет два состояния: зеленый и красный. Каждую секунду значение на циферблате уменьшается на единицу, а после цифры "один" цвет индикатора меняется на противоположный, значение на циферблате сбрасывается.

Мы имеем дело с подтипом таких светофоров:

- Он имеет два разряда на циферблате.
- Он не ведет отсчёт, пока индикатор горит красным цветом.
- Он старый. Некоторые (или все) секции циферблата могут не работать (всегда не гореть). Заранее не известно, какие именно.

Задача наблюдателя - как можно раньше определить (по возможности, до начала первой красной индикации):

- С какого значения начался обратный отсчёт зелёного цвета.
- Какие секции светофора точно не работают.

У наблюдателя есть сервис, в который он каждую секунду отправляет данные:

- Цвет индикатора.
- Информацию о горящих секциях на циферблате (если цвет зелёный).

Его наблюдения заканчиваются в каждом из следующих случаев:

- Он отправил данные в сервис. Сервис ответил, что точно знает, с какого именно значения начался обратный отсчёт.
- Он отправил данные в сервис о том, что только что загорелся красный цвет и дождался ответа.
- Он отправил данные в сервис. Сервис оповестил об ошибке в наблюдениях.

Наблюдатель достаточно умен, чтобы уметь следующее:

- Если в момент начала наблюдения на светофоре горит красный индикатор, наблюдатель просто стоит и ждёт начала зелёной индикации. И только после начала зелёной индикации он начинает отправлять в сервис данные.
- Используя часы, наблюдатель отправляет данные о наблюдениях за каждую секунду, без пропусков и повторений. Даже если все секции циферблата не работают и он не может определить смену состояний на глаз.

2. Задача

Необходимо написать сервис, обрабатывающий запросы от наблюдателя.

2.1. Кодирование цифр

На полностью рабочем циферблате каждая из двух секций отображает цифры следующим образом:

0	1	2	3	4	5	6	7	8	9
-		-	-		-	-	-	-	-
		-	-	-	-	-		-	-
-		-	-		-	-		-	-

В сервис отправляются данные о каждой секции каждого из разрядов циферблата. Секции кодируются следующим образом:

-0
1 2
-3
4 5
-6

Таким образом, если на исправном циферблате горит цифра “4”, данные о секции будут следующими: ‘0111010’. А цифра “7” будет представлена в виде ‘1010010’

2.3. Запросы наблюдателя

Перед началом отправки данных каждый наблюдатель делает запрос в сервис на предоставление уникального кода его текущей последовательности. Последующие данные о наблюдениях сопровождаются этим кодом.

В каждом наблюдении указывается имя последовательности, цвет индикатора и информация о разрядах на циферблате (если цвет зелёный).

2.3. Ответы сервиса

В ответ на запрос о создании последовательности сервис выдаёт уникальный код, с которым в последующем должны быть связаны все наблюдения.

В ответ на каждое наблюдение сервис даёт ответ, содержащий следующую информацию:

- Список цифр, с которых могло быть начато наблюдение.
- Информация о секция светофора, которые однозначно не работают.

Сервис может информировать пользователя об ошибке, отправляя соответствующий статус ответа и сообщение об ошибке.

3. API

3.1. Пример

Рассмотрим пример, когда на циферблате сломаны секции 0 и 5 во втором разряде и наблюдатель подходит в момент, когда должно высветиться значение “02”.

Наблюдатель подходит к светофору и наблюдает на циферблате значение. Сначала он отправляет запрос на создание последовательности.

Запрос:

```
POST /sequence/create HTTP/1.1 {}
```

Ответ:

```
{'status': 'ok', 'response':  
  {'sequence': 'b839e67c-d637-4afc-9241-63943c4fea83'}}
```

Наблюдатель отправляет данные о своём первом наблюдении.

Запрос:

```
POST /observation/add HTTP/1.1 {  
  'observation': {  
    'color': 'green',  
    'numbers': ['1110111', '0011101']  
  },  
  'sequence': 'b839e67c-d637-4afc-9241-63943c4fea83'  
}
```

Ответ:

```
{'status': 'ok', 'response':  
  {'start': [2, 8, 82, 88], 'missing': ['0000000', '1000000']}}
```

Наблюдатель видит на циферблате следующее значение.

Запрос:

```
POST /observation/add HTTP/1.1 {  
  'observation': {  
    'color': 'green',  
    'numbers': ['1110111', '0010000']  
  },  
  'sequence': 'b839e67c-d637-4afc-9241-63943c4fea83'  
}
```

Ответ:

```
{'status': 'ok', 'response':  
  {'start': [2, 8, 82, 88], 'missing': ['0000000', '1000010']}}
```

Наблюдатель видит красную индикацию светофора.

Запрос:

```
POST /observation/add HTTP/1.1 {  
  'observation': {'color': 'red'},  
  'sequence': 'b839e67c-d637-4afc-9241-63943c4fea83'  
}
```

Ответ:

```
{'status': 'ok', 'response':  
  {'start': [2], 'missing': ['0000000', '1000010']}}
```

Наблюдатель получает информацию, что его обратный отсчёт начался со значения “02” и были однозначно сломаны 0-я и 5-я секции второго разряда циферблата.

3.2. Обработка ошибок

В случае ошибки сервис отвечает следующим образом:

```
{'status': 'error', 'msg': <message>}
```

Необходимо сделать обработку ошибок (курсивом - значение поля <message>):

- *“No solutions found”* - не удаётся найти подходящее решение
 - Были пропущены значения
 - Значения были по ошибке отправлены дважды
 - Отправлены не последовательные значения
- *“The sequence isn't found”* - последовательность не найдена или не была создана
- *“There isn't enough data”* - первое же наблюдение сигнализирует о красном цвете индикатора
- *“The red observation should be the last”* - была попытка отправить данные после наблюдения красного индикатора

Также необходимо делать валидацию входных данных и выдавать ошибку в случае несовпадения формата.

3.3. Очистка данных

Дополнительно необходимо реализовать метод очистки сервиса от всех имеющихся в нем данных: всех последовательностях и всех наблюдениях.

Запрос:

```
GET /clear HTTP/1.1
```

Ответ:

```
{'status': 'ok', 'response': 'ok'}
```

4. Требования

4.1. Общие требования

1. Список возможных языков: C#.
2. Сервис должен отвечать на HTTP запросы.
3. API должен в точности соответствовать выше описанному.
4. Сервис должен уметь сигнализировать об ошибках.
5. В проект необходимо включить инструкции по его сборке, установке, настройке, запуску сервиса и тестов.
6. Максимальное время на ответ сервиса - 1 секунда.
7. Возможность легко остановить или запустить сервис без потери данных.
8. Необходимо обеспечить адекватное покрытие unit-тестами.

4.2. Оптимизация

Оптимизация, по степени уменьшения важности:

1. Как можно раньше найти, с какого числа был начал отсчёт.
2. Как можно точнее определить, какие секции точно сломаны.
3. Отсутствие деградации производительности при большом количестве последовательностей и наблюдений.

5. Дополнительные требования

Реализация этих требований **не** обязательна, но будет плюсом:

1. Включить в проект процедуру для оценки эффективности алгоритма.
2. Включить в проект процедуру для оценки производительности сервиса.
3. Написать приёмочные (acceptance) тесты.