Задачи за задължителна самоподготовка

по

Структури от данни и програмиране

email: kalin@fmi.uni-sofia.bg

9 януари 2018 г.

1. (решена) Към разработения на лекции клас `DFSA` да се добави итератор за състояния (`StatesIterator`), чрез който да могат да бъдат обхождани състоянията на автомата, например по следния начин:

```
for (uint state : A)
{
   std::cout << "State label =  "
             << state
             << std::endl;
}
```

2. (решена) Към структурата `state` на разработения на лекции клас `DFSA` да се добави итератор за символи (`SymbolsIterator`), чрез който да могат да бъдат обхождани всички изходящи преходи от състоянието, например по следния начин:

```
for (uint state : A)
{
   for (char symbol : A[state])
   {
       std::cout << "Transition="
                 << state
                 << ":"
                 << symbol
```

```
                    << "->"
                    << A[state][symbol]
                    << std::endl;
}
```

3. (решена) С така създадените методи за достъп да се релизира печатане в `Dotty` формат.

4. (ренеша) Да се подобри класа `DFSA`, така че да могат да бъдат обхождани преходите на константен автомат.

   Упътване: Двата итератора трябва да осигурявят константен достъп. По-особената част е за всеки от операторите за индексиране `[]` (на автомата и на състоянието) да се добави константна версия. Обърнете внимание на разликата между оператор `map::operator []` и метода `map::at(...)`. Случаят е най-интересен за константната версия на `state::operator[]`, за който не се налага proxy.

5. (решена) При условие, че в автомата няма цикли, да се намери най-дългата дума в езика му.

6. (не е решена) Да се намери най-дългата крайна дума в езика на автомат, в който може да има цикли.

7. (не е решена) Да се намери броя на крайните думи в езика на автомат, в който може да има цикли.