# 03 Автодополнение

#### Jump to bottom

Alexander Morozov edited this page 16 days ago · 9 revisions

Ссылка на задание: https://classroom.github.com/a/X\_MWiPbQ

# Задание

Напишите реализацию автодополнения для N строк и положительных весов. На выход программе нужно выдать все строки, которые начинаются с заданного префикса в порядке убывания весов.

Автодополнение очень чувствительно к скорости работы. Поэтому автодополнение нужно реализовать на отсортированном множестве, а поиск в этом множестве нужно делать бинарным поиском.

Задание состоит из трёх частей:

#### Часть 1

Нужно реализовать неизменяемый класс term, который содержит в себе:

- Конструктор принимающий строку и её вес
- Операторы сравнения термов
- Конструкторы/операторы копирования, присваивания, перемещения должны работать корректно
- Метод to\_string который возвращает строковое представление строки в формате "weight term"
- Оператор вывода в поток (тот же формат, что и результат to\_string)
- Meтoд by\_reverse\_weight\_order который возвращает функтор принимающий на вход два терма и сравнивает их с учётом весов в обратном порядке

• Meтод by\_prefix\_order(int r) который возвращает функтор принимающий на вход два терма и сравнивает их по первым r символам без учёта весов

#### Часть 2

Нужно реализовать класс binary\_search\_deluxe, который реализует бинарный поиск и удовлетворяет требованиям:

- Экземпляр этого класса нельзя создавать
- Статический метод int first\_index\_of(term[] a, term key, Func comparator), который принимает отсортированный массив а и должен выдать индекс начала key в массиве а
- Статический метод int last\_index\_of(term[] a, term key, Func comparator), который принимает отсортированный массив а и должен выдать индекс конца key в массиве а

Func comparator - это некоторый объект (класс с переопределённым оператором круглые скобки, функция, лямбда), который прнимает два терма и возращает true если первый терм меньше второго. Пример такого компаратора:

```
auto cmp = [](const term& t1, const term& t2) { return t1 > t2; };
```

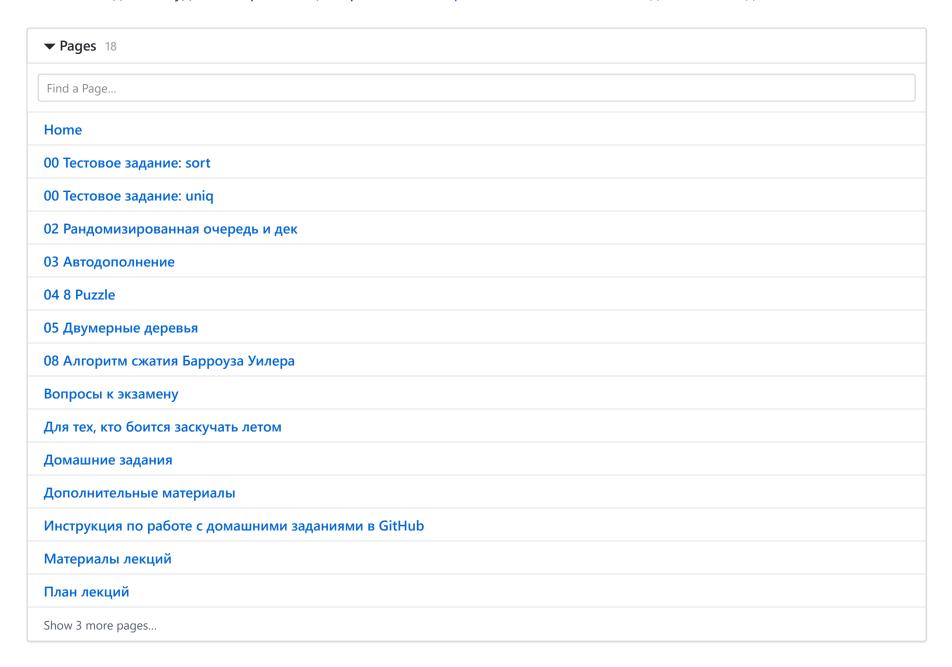
Вместо Func Вам нужно написать такой тип для которого случаи: класс с переопределённым оператором круглые скобки, функция и лямбда будут компилироваться и работать корректно.

## Часть 3

Нужно реализовать неизменяемый класс autocomplete, который использует внутри себя term и binary\_search\_deluxe. Этот класс должен содержать в себе следующие методы:

- Конструктор, принимающий массив термов
- Meтog all\_matches(string prefix) возвращает массив термов, которые подходят под заданный префикс
- Meтoд number\_of\_matches(string prefix) возвращает количество термов, которые подходят под заданный префикс

- 1. Для сигнатур методов нужно выбрать правильные модификаторы доступа
- 2. Работа должно удовлетворять общим требованиям Требования к выполнению домашних заданий



### Clone this wiki locally

https://github.com/itiviti-cpp/wiki.wiki.git

