

### Задача 3. Анализ графа пользователей Twitter

Сообщество пользователей сервиса микроблогов Twitter можно представить в виде ориентированного графа. Наличие дуги  $(i, j)$  в графе означает, что пользователь  $i$  является подписчиком пользователя  $j$  (follower).

На рис. 1 показан пример графа подписчиков: пользователь Макар подписан на микроблог Авдотьи и Игната, а на микроблог Макара подписан Евдоким.

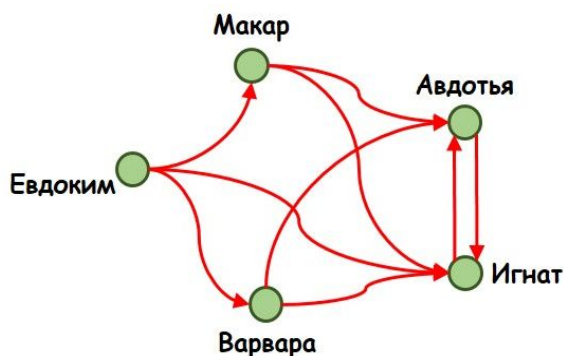


Рис. 1. Пример графа подписчиков (followers)

Граф пользователей с информацией о их подписчиках хранится в текстовом файле, каждая строка которого имеет следующий формат:

```
<user_id> <follower_id>
```

Файл размещен на кластере:

```
/home/ipa/pub/hadoop/lab34/followers.db
```

#### Требуется

- Вычислить распределение количества подписчиков (статистическое распределение выборки)
- Определить Top50 пользователей по числу подписчиков
- Вычислить среднее количество подписчиков

Программы должны быть написаны на языке Java (Apache Hadoop Java API).

## Задача 4. Инвертированный индекс Wikipedia

Требуется построить инвертированный индекс (inverted index) для русской Википедии (и английской — по желанию).

Дампы Википедии в формате XML загружены на кластер Jet: (<http://dumps.wikimedia.org>):

```
/home/pub/hadoop/lab34/enwiki-*
```

```
/home/pub/hadoop/lab34/ruwiki-*
```

Дамп — это большой XML-файл, в котором каждая Wiki-статья размечена парой тегов `<page>[ARTICLE]</page>`. Статья имеет числовой идентификатор (`<id>[ID]</id>`) и текстовое содержание (Wiki-разметка, `<text>[TEXT]</text>`).

Файл русской Википедии занимает > 10 GiB, файл английской Википедии > 40 GiB.

Для работы с XML-дампом нужна специальная версия класса `InputFormat`, которая будет разбирать фрагмент XML-файла (`Split`) и передавать функции `map` в качестве ключа идентификатор статьи (`docid`) и текст статьи в качестве значения (`content`).

На сайте приведен пример реализации класса `XmlInputFormat` (это модификация `XmlInputFormat` из Apache Mahout).

Входные данные `map`:

```
(docid, content)
```

Результирующий инвертированный индекс должен иметь следующую структуру:

```
(word, [<docid1, TF-IDF1>, <docid2, TF-IDF2>, ...])
```

- Статьи должны быть отсортированы в порядке убывания TF-IDF (Term Frequency – Inverse Document Frequency)
- Для каждого слова ограничить список статей  $N$  наиболее релевантными
- Определить и исключить из индекса Top20 высокочастотных слов

При вычислении TF-IDF считаем, что:

- $TF(t, d)$  — это число вхождений слова  $t$  в документ  $d$  (Wiki-статью)
- $IDF(t, D)$  — обратная частота, с которой слово  $t$  встречается во множестве документов  $D$  (Wiki-статьях):

$$IDF(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Программы должны быть написаны на языке Java (Apache Hadoop Java API).