作业3

> 姓名： 吴双
>
> 学号： 10164102141

1 软件体系结构

WordSort

> 要求
>
> > 将之前的WordCount改为WordSort排序
>
> 思路
>
> > 同样，使用一个list来记录当前位置，同时使用二分查找找到当前位置并插入

top-k

> 要求
>
> > 单词的top-k：求最频繁的k个word 要考虑代码的性能
>
> 思路
>
> > 首先仍然与WordCount一样，需要一个 hashmap 来存当前word对应的count数量 考虑到性能原因，只需要维护前k个word即可 需要设置最小count，表示前k个中最小的count数，如果当前有word的count数大于这个数，则将其加入TopK数组，然后剔除

2 详细设计

Wordsort:

SentenceSpout

```
public class SentenceSpout extends BaseRichSpout {

    private SpoutOutputCollector spoutOutputCollector;
    private String[] sentences = {"the cow jumped over the moon", "an apple a day keeps the doctor away",
```

```java
            "four score and seven years ago", "snow white and the seven dwarfs", "i am at two with nature"};

    public void open(Map map, TopologyContext topologycontext, SpoutOutputCollector spoutoutputcollector) {
        this.spoutOutputCollector = spoutoutputcollector;
    }

    public void nextTuple() {
        for (String sentence : sentences) {
            Values values = new Values(sentence);
            UUID msgId = UUID.randomUUID();
            this.spoutOutputCollector.emit(values, msgId);
        }
        Utils.sleep(1000);
    }

    public void declareOutputFields(OutputFieldsDeclarer outputfieldsdeclarer) {
        outputfieldsdeclarer.declare(new Fields("sentence"));
    }

}
```

SplitSentenceBolt

```java
public class SplitSentenceBolt extends BaseBasicBolt {
    public void execute(Tuple tuple, BasicOutputCollector collector) {
        String sentence = tuple.getStringByField("sentence");
        String[] words = sentence.split(" ");
        for (String word : words) {
            collector.emit(new Values(word));
        }
    }

    public void declareOutputFields(OutputFieldsDeclarer outputfieldsdeclarer) {
        outputfieldsdeclarer.declare(new Fields("word"));
    }

}
```

WordSortBolt

```java
public class WordSortBolt extends BaseBasicBolt {
    List<String> wordList = new ArrayList<String>();

    public int arrayIndexOf(String key) {
        int min, max, mid;
        min = 0;
        max = wordList.size() - 1;

        while (min <= max) {
            mid = (min + max) >> 1;
            String tmp = wordList.get(mid);
            if (key.compareTo(tmp) > 0) {
                min = mid + 1;
            } else if (key.compareTo(tmp) < 0) {
                max = mid - 1;
            } else {
                return mid;
            }
        }
        return min;
    }


    public void execute(Tuple tuple, BasicOutputCollector collector) {
        String word = tuple.getString(0);
        if (wordList == null) {
            wordList.add(word);
            System.out.println(word);
        } else {
            int addIndex = arrayIndexOf(word);
            wordList.add(addIndex,word);
            for (String tmp:wordList
                 ) {
                System.out.println(tmp);
            }
        }
        collector.emit(new Values(word));
    }

    public void declareOutputFields(OutputFieldsDeclarer outputfieldsdeclarer) {
        outputfieldsdeclarer.declare(new Fields("word"));
    }
}
```

WordSortTopology

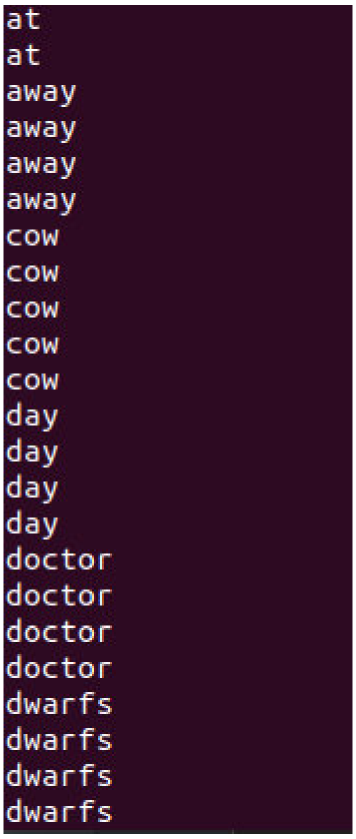
```java
public class WordSortTopology {

    public static void main(String[] args) throws Exception {

        SentenceSpout sentenceSpout = new SentenceSpout();
        SplitSentenceBolt splitSentenceBolt = new SplitSentenceBolt();
        WordSortBolt wordSortBolt = new WordSortBolt();

        TopologyBuilder builder = new TopologyBuilder();
        builder.setSpout("sentenceSpout-1", sentenceSpout);
        builder.setBolt("splitSentenceBolt-1",
splitSentenceBolt).shuffleGrouping("sentenceSpout-1");
```

```
        builder.setBolt("wordSortBolt-1", 
wordSortBolt).fieldsGrouping("splitSentenceBolt-1", new Fields("word"));

        Config config = new Config();
        LocalCluster cluster = new LocalCluster();

        cluster.submitTopology("wordSortTopology-1", config, 
builder.createTopology());
        Thread.sleep(999999999);
        cluster.shutdown();
    }

}
```

> 将程序打包成JAR包，并shell input:

```
storm jar WordSort.jar WordSortTopology ws
```

> output:



```
at
at
away
away
away
away
cow
cow
cow
cow
cow
day
day
day
day
doctor
doctor
doctor
doctor
dwarfs
dwarfs
dwarfs
dwarfs
```

K-top:

SentenceSpout

```
public class SentenceSpout extends BaseRichSpout {

    private SpoutOutputCollector spoutOutputCollector;
```

```java
    private String[] sentences = {"the cow jumped over the moon", "an apple a day keeps the doctor away",
            "four score and seven years ago", "snow white and the seven dwarfs", "i am at two with nature"};

    public void open(Map map, TopologyContext topologycontext, SpoutOutputCollector spoutoutputcollector) {
        this.spoutOutputCollector = spoutoutputcollector;
    }

    public void nextTuple() {
        for (String sentence : sentences) {
            Values values = new Values(sentence);
            UUID msgId = UUID.randomUUID();
            this.spoutOutputCollector.emit(values, msgId);
        }
        Utils.sleep(1000);
    }

    public void declareOutputFields(OutputFieldsDeclarer outputfieldsdeclarer) {
        outputfieldsdeclarer.declare(new Fields("sentence"));
    }

}
```

SplitSentenceBolt

```java
public class SplitSentenceBolt extends BaseBasicBolt {
    public void execute(Tuple tuple, BasicOutputCollector collector) {
        String sentence = tuple.getStringByField("sentence");
        String[] words = sentence.split(" ");
        for (String word : words) {
            collector.emit(new Values(word));
        }
    }

    public void declareOutputFields(OutputFieldsDeclarer outputfieldsdeclarer) {
        outputfieldsdeclarer.declare(new Fields("word"));
    }

}
```

pair

```java
public class pair {
    public final String content;
    public final Integer count;
    public pair(String content, Integer count) {
        this.content = content;
        this.count = count;
    }

    public int compareCount(pair other) {
        return this.count - other.count;
    }

    public int compareWord(pair other) {
        return this.content.compareTo(other.content);
    }
}
```

TopK 使用HashMap来保存当前所有word的count数量，使用TopList保存前k个word

```java
public class TopK extends BaseBasicBolt {
    private HashMap<String, Integer> counts;
    private ArrayList<pair> TopList;
    public int K;
    public int minCount;

    TopK(int k) {
        this.K = k;
        this.counts = new HashMap<>();
        this.TopList = new ArrayList<>();
        this.minCount = 0;
    }


    public void insertWord(pair word) {
        int max = TopList.size() - 1;

//        if the same word, the new.count > old.count
        for (int i = max; i >= 0; i--) {
            pair tmp = TopList.get(i);
            // find the same word,replace older one
            if (word.compareWord(tmp) == 0) {
                TopList.set(i, word);
                return;
            }
            if (word.compareCount(tmp) <= 0) {
                TopList.add(i + 1, word);
                return;
            }
        }
        TopList.add(0, word);
    }


    public void execute(Tuple tuple, BasicOutputCollector collector) {
        String word = tuple.getStringByField("word");
        Integer count = counts.get(word);
        if (count == null) {
            count = 0;
        }
        count++;
        this.counts.put(word, count);
        if (count > minCount || TopList.size() < K) {
            insertWord(new pair(word, count));
            if (TopList.size() > K) {
                TopList.remove(TopList.size() - 1);
                minCount = TopList.get(TopList.size() - 1).count;
            }
            for (pair tmpWord : TopList)
                System.out.println(tmpWord.content + " " + tmpWord.count.toString());
            collector.emit(new Values(word, count));
        }


    }

    public void declareOutputFields(OutputFieldsDeclarer outputfieldsdeclarer) {
        outputfieldsdeclarer.declare(new Fields("word", "count"));
    }
}
```

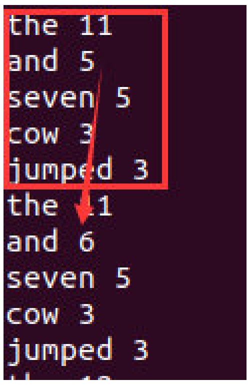
WordTopKTopology

```java
public class WordTopKTopology {
    public static void main(String[] args) throws Exception {

        SentenceSpout sentenceSpout = new SentenceSpout();
        SplitSentenceBolt splitSentenceBolt = new SplitSentenceBolt();
        TopK wordTopKBolt = new TopK(5);

        TopologyBuilder builder = new TopologyBuilder();
        builder.setSpout("sentenceSpout-1", sentenceSpout);
        builder.setBolt("splitSentenceBolt-1", splitSentenceBolt).shuffleGrouping("sentenceSpout-1");
        builder.setBolt("wordTopKBolt-1", wordTopKBolt).shuffleGrouping("splitSentenceBolt-1");

        Config config = new Config();
        LocalCluster cluster = new LocalCluster();

        cluster.submitTopology("wordTopKTopology-1", config, builder.createTopology());
        Thread.sleep(999999999);
        cluster.shutdown();
    }
}
```

shell input:

```
storm jar TopK.jar WordTopKTopology tk
```

> output:



```
the 11
and 5
seven 5
cow 3
jumped 3
the 11
and 6
seven 5
cow 3
jumped 3
```