场景题

海量数据处理

- 前缀树
  - 基本性质
  - 代码实现
  - 查询复杂度
  - 应用场景

TopK问题

- 思路
- 典型例题

重复元素问题

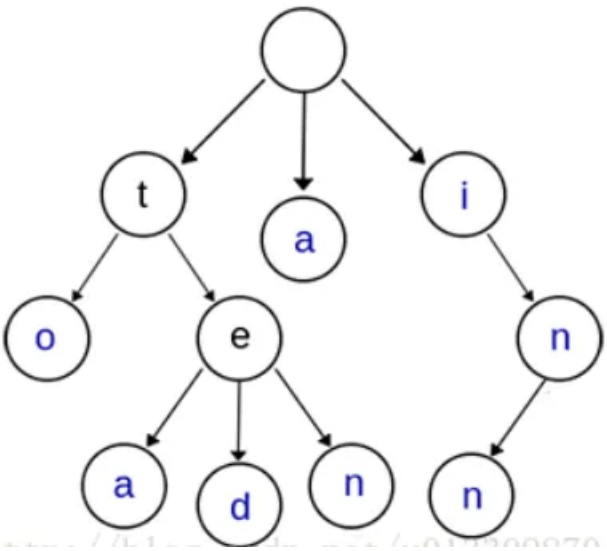
- 思路
- 典型例题

海量数据排序

- 思路

海量数据处理

前缀树



基本性质

- 除根节点外每个节点都包含字符
- 从根节点到某个节点路径的所有字符串联起来形成该节点对应的字符串
- 每个节点的所有子节点包含的字符各不相同，且这些字符是按照字典序排列的

代码实现

```java

public class Trie {

    private class Node{

        public boolean isWord; // 是否是某个单词的结束

        public TreeMap<Character, Node> next; //到下一个节点的映射

        public Node(boolean isWord){
            this.isWord = isWord;
            //初始化字典树
            next = new TreeMap<>();
        }

        public Node(){
            this(false);
        }
    }

    //根节点
    private Node root;
    //Trie单词个数
    private int size;

    public Trie(){
        root = new Node();
        size = 0;
    }

    // 获得Trie中存储的单词数量
    public int getSize(){
        return size;
    }

}

```

查询复杂度

时间复杂度：o(n)

应用场景

1. 词频统计（每个节点记录对应字符串出现次数）
2. 字符串排序（每个节点子节点按照顺序排列，上述用得TreeMap）
3. 字符串检索
4. 前缀匹配

TopK问题

在海量数据（无法一次性读入内存）中，寻找出现次数最多的K个元素。

思路

分治+HashMap/TrieTree+堆/快速排序

1. 将海量数据根据关键值hash到M个文件中；
2. 依次对M个文件用hashmap统计各元素出现的次数，然后取出现次数最多的K个元素，将元素和对应出现次数存入内存；
3. 将上述MK个元素和按照其出现次数进行排序，最终得到TopK个元素

典型例题

Q：有一个1G大小的一个文件，里面每一行是一个词，词的大小不超过16字节，内存限制大小是1M。返回频数最高的100个词。

A：

1. 将1G文件按照词划分到5000个小文件中（那么每个文件大约200K，若有文件超过1M，继续将其划分，分成大小小于200K的文件）
2. 分别统计上述这些小文件中每个词出现的次数，将每个文件中出现次数Top100的词放在内存中
3. 将上述所有小文件的Top100词按照出现频率进行堆排序，得到最终结果

重复元素问题

在海量数据中，找出重复/不重复的数据；多个文件的共同元素。

思路

使用bloomfilter，或者内存允许的话直接用bitmap

1. 使用bloomfilter：取内存中一定的bit位，作为bitmap；使用k个hash函数，将每个元素散列到k个bit上；若某个元素的k个bit均已经为1，则认为该元素存在/重复；反之该元素不重复
2. 使用bitmap：已知海量元素的最大值为N，取一个N位的bitmap，遍历所有元素，将其对应的bit设置为1

典型例题

Q：给40亿个不重复的unsigned int的整数，没排过序的，然后再给一个数，如何快速判断这个数是否在那40亿个数当中?

A：

1. unsigned int有32bit，最大值是2^32-1；取2^32-1个bit，即512MB；
2. 将40亿个元素对应的数的bit设置为1；
3. 检查当前元素对应的bit是否为1即可

Q：在2.5亿个整数中找出不重复的整数，注，内存不足以容纳这2.5亿个整数。

A：

1. int有32bit，用2bit表示元素是否存在（00不存在，01存在，再来就10，其他也是10），共需要2^32*2bit=1GB内存
2. 遍历所有元素，设置对应的2-bit位
3. 所有2-bit值为01的即为结果

海量数据排序

将海量数据（不能全部放入内存）（重复或者非重复）进行排序。

思路

1. 若没有重复元素，在知道元素取值范围的情况下，用bitmap，将每个元素对应的bit位设置为1，最后按顺序将这些数据写入文件即可
2. 若有重复元素，不能使用bitmap，使用外排序：
    a. 将海量数据划分到M个小文件中
    b. 每个文件依次读入内存，进行排序，将排序后的结果写回文件
    c. 排序时，每次从M个文件中读取最小值，取M个最小值作为当前排序确定的元素（同时记录M个元素的最小值对应的文件index，取他的下一个最小元素进来，作为下一次比较的候选值）