近似查询

计64 翁家翌 2016011446

效果

OJ submission ID: 7774

ID	Homework	Upload Timestamp	Status	Memory(GB)	Time(s)	Comment	Ops
7774(Marked)	exp1-final	2019/04/14 16:44:22	Running.	8.247	5.975	no bin	✖

实现

初始化

对于ED，仅使用Trie建立倒排表

对于Jaccard（以下简称JC），同时使用Trie和hash，其中Trie建立倒排表，hash用来最后判断是否合法

由于该部分不计入时间，因此未对其做特殊优化。

searchED

1. 计算 T 的下界，公式为

$$T_{min} = |Q| + 1 - (\tau + 1) \times q$$

   如果 $T_{min} \leq 0$，则需遍历所有字符串并进行一一判断。
2. 遍历询问字符串，在Trie中查找对应倒排表。记总个数为 $token\_len$。查询完毕之后将结果按照每个倒排表长度升序排序。
3. 设置shortlist个数，默认为 $token\_len - (T_{min} - 1)$，使用scancount统计各个候选字符串出现次数。
4. 在3中筛选出的候选字符串中，依次判断是否合法：在longlist中二分，计算它在筛出的倒排表中出现总次数，如果出现次数 $\geq T_{min}$，则进入 `queryED` 环节。如果 `queryED` 返回成功，那么将其添加到答案中。
5. 将答案排序并返回。

queryED(s, t)

这是个动态规划问题，记 $f_{i,j}$ 表示 s[0:i], t[0:j] 两个字符串最少需要的编辑次数，边界条件如下：

$$f_{0,i} = f_{i,0} = i$$

转移方程如下：

$$f_{i,j} = \min(f_{i-1,j} + 1, f_{i,j-1} + 1, f_{i-1,j-1} + 1[s_{i-1} \neq t_{j-1}])$$

$f_{|s|,|t|}$ 即为所求，时间复杂度为 $O(|s||t|)$。

searchJC

大致步骤同 `searchED`，除了把对应接口从ED换成JC之外，唯一区别如下：

记字符串出现的单词数为 num，数据中一行最少出现的单词数为 num_{min}，T 的下界为

$$\lceil \max(num \times \tau, (num + num_{min}) \times \frac{\tau}{1+\tau}) \rceil$$

queryJC(s, t)

s和t均为将原始字符串的单词hash之后的数组，长度为单词个数，按照升序排序。

问题规约成，有两个数组，均为升序排序，数出其中相同元素的个数。

按顺序扫一遍即可。时间复杂度为 $O(\max(|s|, |t|))$。

优化

q的选取

实测q越大跑越快，q=8为上限，再大就爆内存了

MergeOpt vs DivideSkip

实测 T-1 最好，T-2 / T-3 / ... 都不如 T-1

Trie + AC自动机

在查询ED的时候，注意到可以使用AC自动机fail指针的思想，将每个字符串q次在Trie上的操作缩短为平均1次操作

松界 vs 紧界

1. 注意到 T_{min} 其实在步骤4的时候可以收的更紧
2. 在步骤4中，如果当前计数+剩下未计数的longlist的结果比 T_{min} 还要小，直接退了，不可能是答案
3. 在步骤4中，如果当前计数超过 T_{min} ，直接到 `query` 环节

针对 ED 的预判

分析一下 `queryED`，如果 $\Big||s| - |t|\Big| > \tau$，则直接判断不可能成为候选。

因此可在预处理的时候复制一份倒排表，按照每个字符串的长度排序id。

在遍历查询的时候（两种情况：$T_{min} \le 0$ 或者 scancount），可以先二分出上下界 L 和 R，满足在这个范围内的id集合所对应的字符串长度处于 $[|Q| - \tau, |Q| + \tau]$ 之间，这样需要遍历的字符串数目可减少一些。

经过尝试，该优化应加在步骤4中而不是步骤3中，虽然从直觉上感觉加在步骤4中应该更快，但是运行效率和数据分布的关系还挺大的。加在步骤3会慢个0.1-0.4s.....

针对 JC 的预判

基本原理同上，此处不再复述。

唯一区别为，可能的字符串单词数目区间满足 $[1, \lceil \frac{num}{\tau} \rceil]$。

DP的优化

注意到，如果 $|i-j|>\tau$，则状态是无效的，不需要转移。因此时间复杂度可减少为 $O(\tau \max(|s|,|t|))$。

内存连续分配

实测无效果。“libc的内存分配还是很厉害的！”

二分?

q=8的时候不需要二分，直接进query，会快1s+