```cpp
#include <map>
#include <string>

class Trie {
public:
  Trie() {
    head = new TrieNode();
  }

  void insert(std::string &world) {
    int len = world.size();
    if (0 == len) {
      return;
    }
    TrieNode *node  = head;
    int       index = 0;

    for (int i = 0; i < len; i++) {
      index = world[i] - 'a';
      if (node->map.count(index) == 0) {
        // 不存在新建节点
        node->map[index] = new TrieNode();
      }

      node = node->map[index];
      node->path++;  // node 经历的path++
    }

    node->end++;  //找到结尾
  }

  // 字典树查找
  bool search(std::string &str) {
    int len = str.size();
    if (len == 0) {
      return false;
    }

    TrieNode *node  = head;
    int       index = 0;

    for (int i = 0; i < len; i++) {
      index = str[i] - '0';
      if (0 == node->map.count(index)) {
        return false;
      }

      node = node->map[index];
    }

    return node->end != 0;
```

```
  }

  void deleteNode(std::string str) {
    if (!search(str)) {
      return;
    }

    int       index = 0;
    TrieNode *node  = head;
    for (int i = 0; i < str.size(); i++) {
      index = str[i] - '0';
      if (node->map[index]->path-- == 1) {
        node->map[index] = nullptr;
        return;
      }

      node = node->map[index];
    }

    node->end--;
  }

  // 返回以str为前缀的单词数量
  int findPrifex(std::string &str) {
    int len = str.size();
    if (0 == len) {
      return 0;
    }

    TrieNode *node  = head;
    int       index = 0;

    for (int i = 0; i < len; i++) {
      index = str[i] - 'a';
      if (node->map.count(index) == 0) {
        return 0;
      }
      node = node->map[index];
    }

    return node->path;
  }

private:
  struct TrieNode {
    int                         path;  // 有多少单词公用这个节点
    int                         end;   // 有多少单词以这个节点结尾
    std::map<int, TrieNode *> map;

    TrieNode() {
      path = 0;
      end  = 0;
    }
  };
```

```
  TrieNode *head;
};
```