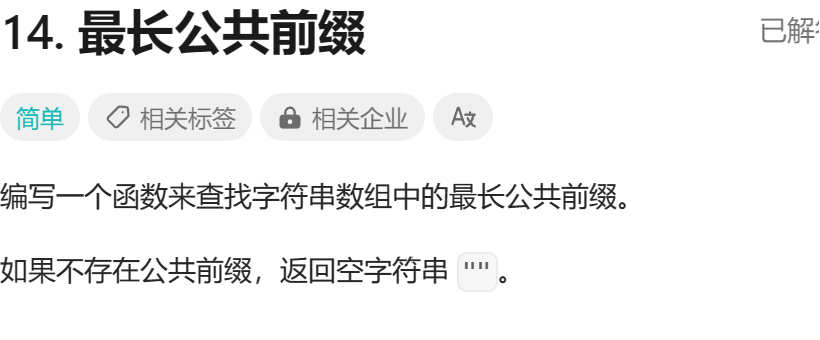
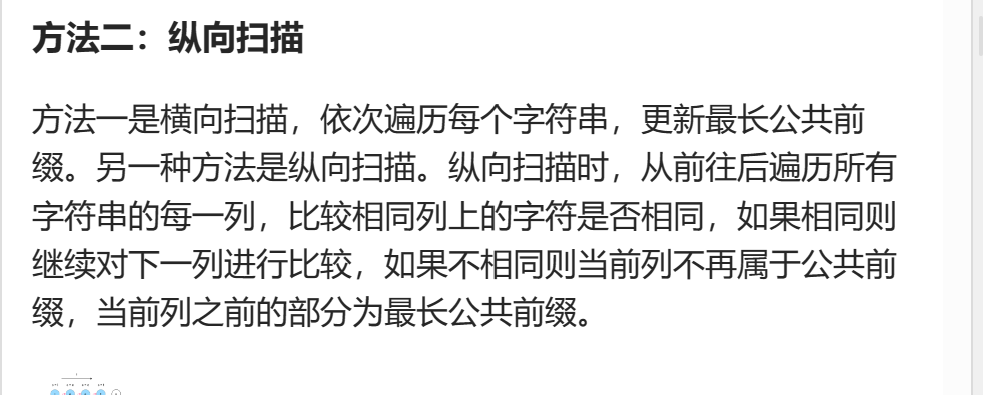
字典树





  string longestCommonPrefix(vector<string>& strs) {

        if(strs.size()==0){

            return "";

        }

        sort(strs.begin(),strs.end(),[&](string a ,string b){ return a.size()<b.size();});

        int xn=strs.size();

        int yn=strs[0].size();

        int len=0;

        for(int i=0;i<yn;i++){

            char tem=strs[0][i];

            for(int j=1;j<xn;j++){

                if(strs[j][i]!=tem){

                    return strs[0].substr(0,len);

                }

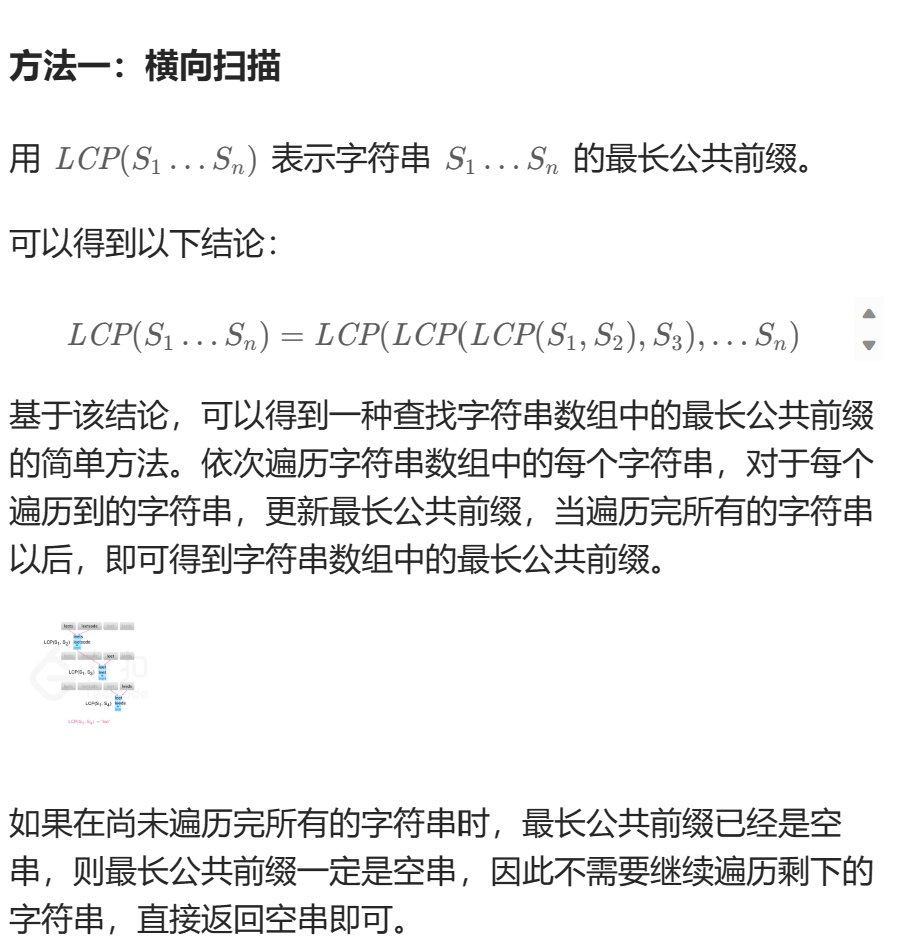
            }

            len++;

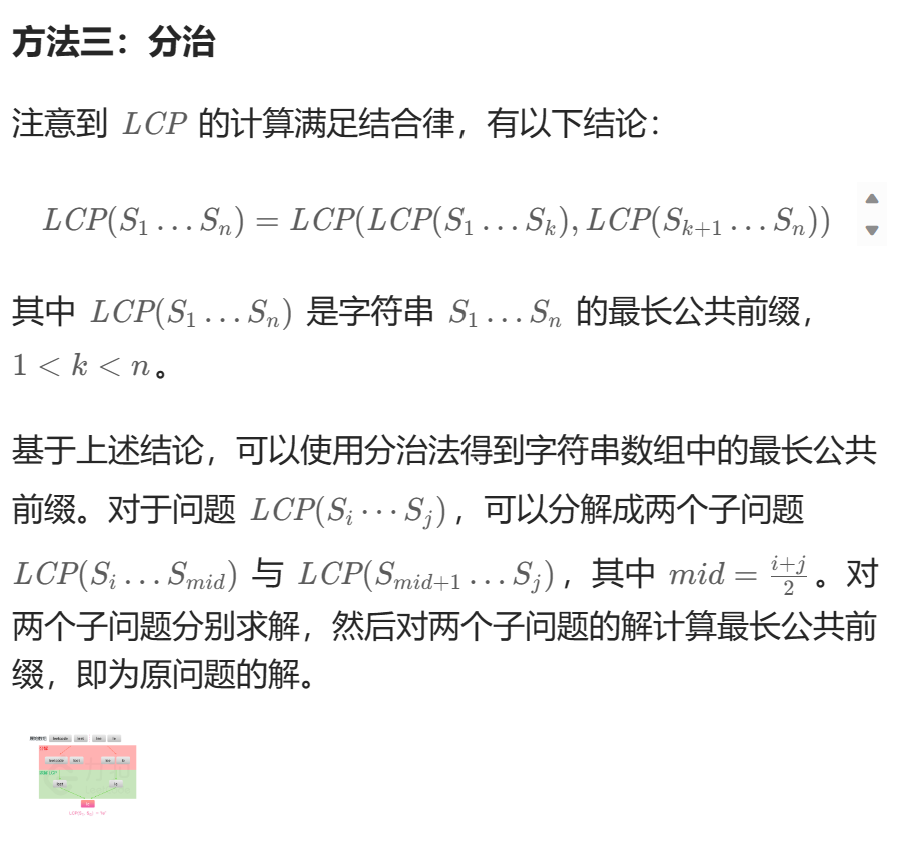
        }

        return strs[0].substr(0,len);

}







 string get\_fix(string &s1,string &s2){

        int len=min(s1.size(),s2.size());

        int  ans=0;

        for(int i=0;i<len&&s1[i]==s2[i];i++){

         ans++;

        }

        return  s1.substr(0,ans);

    }

    string mergy(vector<string>& strs,int lef,int rig){

        if(lef==rig){

            return strs[lef];

        }

          else{

              int mid=(lef+rig)/2;

           string s1=mergy(strs,lef,mid);

           string s2=mergy(strs,mid+1,rig);

           return get\_fix(s1,s2);

          }

    }

    string longestCommonPrefix(vector<string>& strs) {

     if(strs.size()==0){

         return "";

     }

     return mergy(strs,0,strs.size()-1);

    }



构建一颗自动树的代码模板

class Trie {

public:

     // 利用this 的特性在当前类上直接构建一颗 字典树

    //字典树 多叉树-》要用到指针数组来存储子树根节点

    vector<Trie\*> children;//C++ 会自动将字符初始化为空指针

    bool isend;

    Trie():children(26),isend(false) {

    }

    Trie \* searchPrefix(string prefix){

        Trie \* node=this;

        for(auto i:prefix){

            i-='a';

            if(node->children[i]==nullptr){

              return nullptr;

            }

            node=node->children[i];//迭代法查找多叉映射树

        }

        return node;

    }

    void insert(string word) {

        Trie \* node=this;

        for(auto i:word){

            i-='a';

            if(node->children[i]==nullptr){

                node->children[i]=new Trie();//new Trie();C++ 允许的写法 分配一Trie 大小的空间 并且调用构造函数 对其进行初始化

            }

            node=node->children[i];

        }

        node->isend=true;

    }

    bool search(string word) {

        Trie \*node =this->searchPrefix(word);

        return node!=nullptr&&node->isend;

    }

    bool startsWith(string prefix) {

            return this->searchPrefix(prefix)!=nullptr;

    }

};