1. Implement Trie (Prefix Tree)

Implement a trie with insert, search, and startsWith methods.

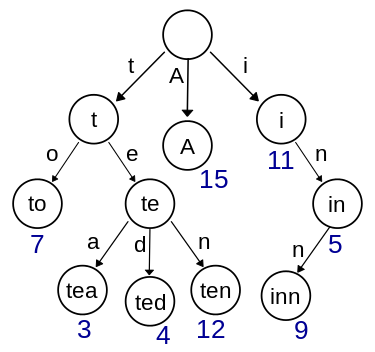
**Example:**

Trie trie = new Trie();  
  
trie.insert("apple");  
trie.search("apple"); // returns true  
trie.search("app"); // returns false  
trie.startsWith("app"); // returns true  
trie.insert("app");   
trie.search("app"); // returns true

**Note:**

* You may assume that all inputs are consist of lowercase letters a-z.
* All inputs are guaranteed to be non-empty strings.

**解**



class trie\_node{  
public:  
 vector<trie\_node\*>child; // 根据当前字符选取下一个节点，如果当前字符是a，接下来就往child['a']的分支走  
 bool isWord; // 表明到达该节点时，是否包含了一个完整的单词  
 // 构造函数  
 trie\_node(): isWord(false), child(26, NULL){}  
 // 析构函数  
 ~trie\_node(){  
 for(auto &c : child)delete c;  
 }  
};  
class Trie {  
public:  
 trie\_node\* root;  
 /\*\* Initialize your data structure here. \*/  
 // 构造函数  
 Trie() {  
 root = new trie\_node();  
 }  
 /\*\* Inserts a word into the trie. \*/  
 void insert(string word) {  
 trie\_node \*cur = root;  
 for(int i = 0; i < word.size(); ++i){  
 int idx = word[i] - 'a';  
 if(cur->child[idx] == NULL){  
 cur->child[idx] = new trie\_node();  
 }  
 cur = cur->child[idx];  
 }  
 cur->isWord = true;  
 }  
   
 /\*\* Returns if the word is in the trie. \*/  
 bool search(string word) {  
 trie\_node \*cur = root;  
 for(auto ch : word){  
 int idx = ch - 'a';  
 if(cur->child[idx] == NULL)return false;  
 cur = cur->child[idx];  
 }  
 return cur->isWord;  
 }  
   
 /\*\* Returns if there is any word in the trie that starts with the given prefix. \*/  
 bool startsWith(string prefix) {  
 trie\_node \*cur = root;  
 for(auto ch : prefix){  
 int idx = ch - 'a';  
 if(cur->child[idx] == NULL)return false;  
 cur = cur->child[idx];  
 }  
 return true;  
 }  
};  
  
/\*\*  
 \* Your Trie object will be instantiated and called as such:  
 \* Trie\* obj = new Trie();  
 \* obj->insert(word);  
 \* bool param\_2 = obj->search(word);  
 \* bool param\_3 = obj->startsWith(prefix);  
 \*/