问题

处理字符串问题时会遇到：

- 查找一个字符串是否是某些字符串的前缀，**查询前缀**
- 统计单词集里哪个单词出现的频率最高，**词频统计**
- …

~~当然这里面部分问题字符串hash也能做~~

信息学

字符串-字典树Trie

西南大学附属中学校

信息奥赛教练组

字典树

Trie，又称前缀树、单词查找树、键树，是一种树形结构，是一种哈希树的变种

一种多模式匹配算法，也有很多数据结构在字典树基础上进行优化改进

主要的操作：

- **插入**操作，就是将一个字符串W加入到集合中。
- **查询**操作，就是查询一个字符串S是不是在集合中。

Trie树的3个基本性质：

1.根节点不包含任何字符

2.从根节点到某一节点，**路径边上经过的字符连接起来**，为该节点对应的字符串

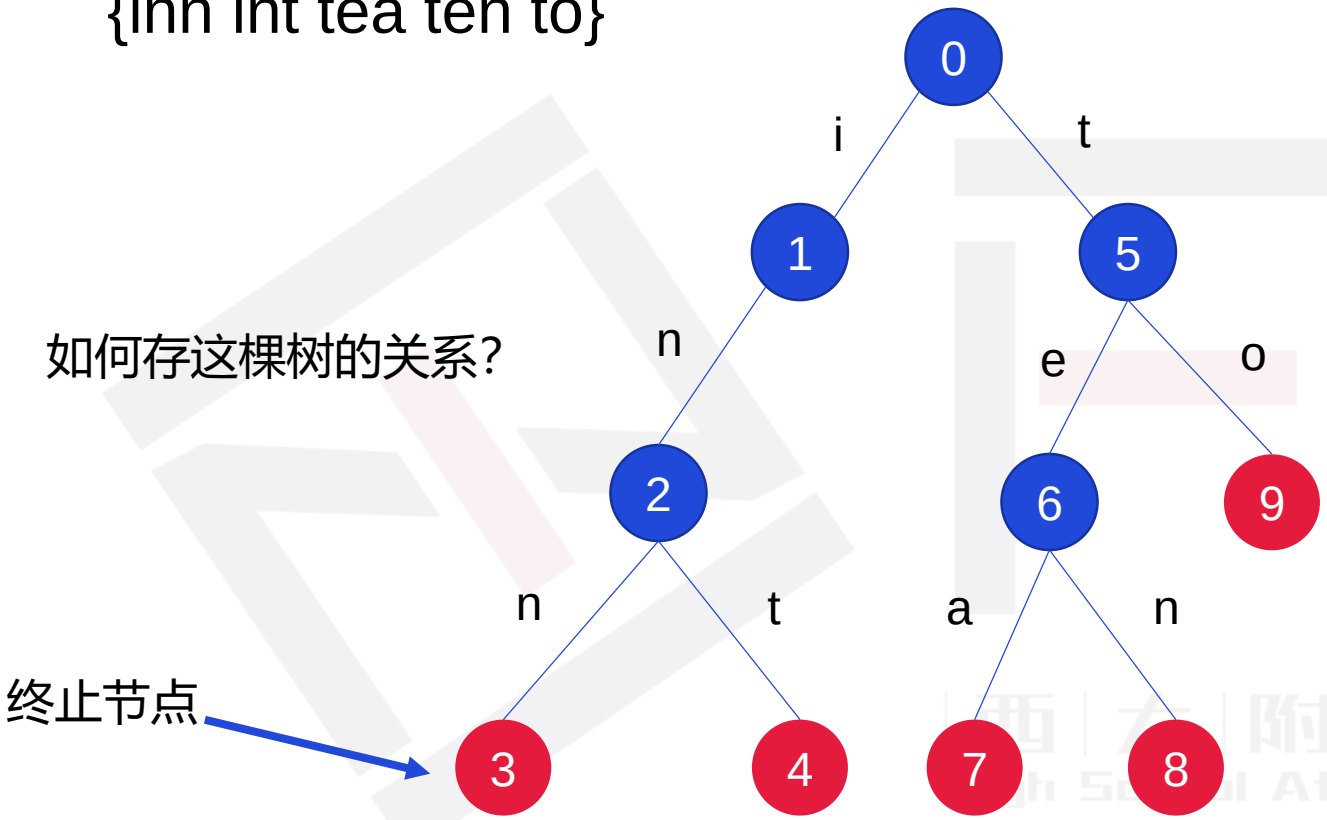
3.每个节点与所有子节点的连接边包含的字符都不相同

字典树在边上存储信息

建树

{inn int tea ten to}

如何存这棵树的关系?



查询too

0->5->9，9号节点无其他节点
不存在

查询tee

0->5->6，6号节点的子节点无e的信息
不存在

查询in

0->1->2，2号节点不是终止节点
不存在

通常字典树的查询时间复杂度是O(logn)，n是字符串的长度。

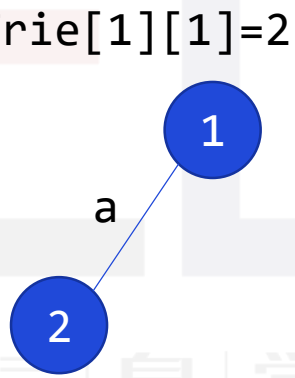
字典树存储


```
int Trie[MAX_NODE][CHARSET];
int nodes;
```

其中MAX_NODE是Trie中最大能存储的节点数目，CHARSET是字符集的大小，nodes是当前Trie中包含有多少个节点。

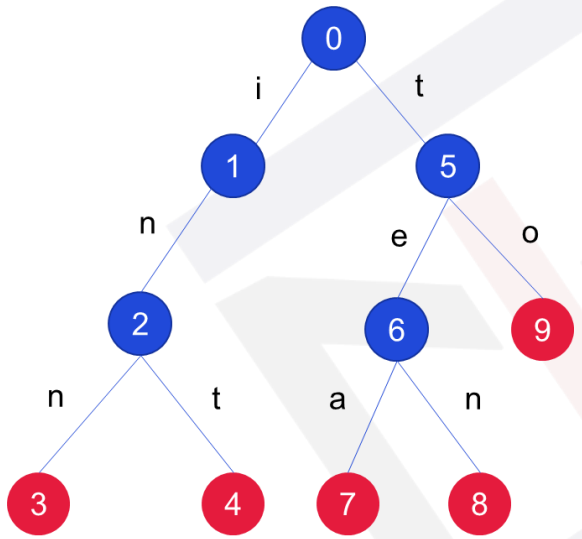
Trie[i][j]：节点i的j字符指向的结点编号

Trie[i][j]的值是0表示trie树中i号节点，没有一条连出去的边,也就是没有子节点



字典树存储


```
int Trie[MAX_NODE][CHARSET];
```



i j	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0									1											5						
1														2												
2														3						4						
3																										
4																										
5					6										9											
6	7													8												
7																										
8																										
9																										

查询一个串是否存在-参考代码


```
#include<bits/stdc++.h>
using namespace std;
const int MAX_NODE = 1000000 + 10;
const int CHARSET = 26;
int trie[MAX_NODE][CHARSET] = {0};
int color[MAX_NODE] = {0};
int nodes = 1;

void insert(char w[]){
    int len = strlen(w);
    int p = 0;
    for(int i=0; i<len; i++){
        int c = w[i] - 'a';
        if(!trie[p][c]) trie[p][c] = nodes++;
        p = trie[p][c];
    }
    color[p] = 1;
}
```

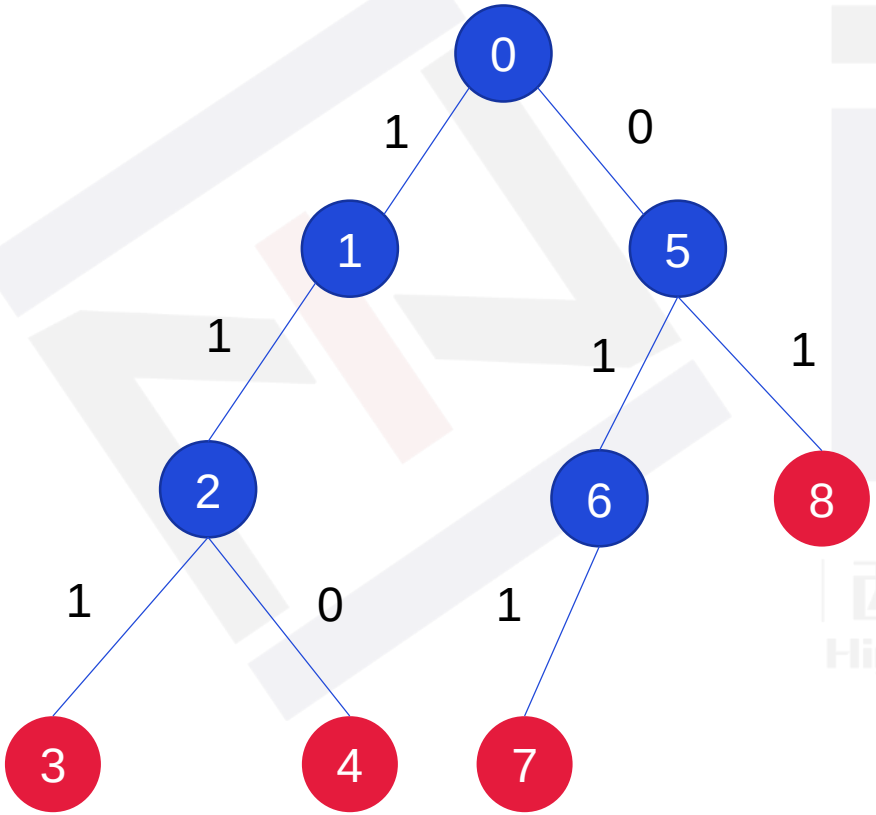
```
int search(char s[]){
    int len = strlen(s);
    int p = 0;
    for(int i=0; i<len; i++){
        int c = s[i] - 'a';
        if(!trie[p][c]) return 0;  //如果不存在
        p = trie[p][c];
    }
    return color[p] == 1;
}

int main(){
    int t,q;
    char s[20];
    scanf("%d%d", &t,&q);
    while(t--){
        scanf("%s", s);
        insert(s);
    }
    while(q--){
        scanf("%s", s);
        if(search(s)) printf("YES\n");
        else printf("NO\n");
    }
    return 0;
}
```

查询和插入的代码很相似
区别在于遇到子结点为空时所做的处理

01字典树

如果需要进行位运算，把一堆整数拆成二进制存储到trie中
就变成了01字典树



例题：最大异或值

例题：最大异或值

在给定的N个整数A1，A2……An中选出两个进行XOR运算，得到的结果最大是多少?

$N<=10^5$, $0<=Ai<2^{31}$

方法一：暴力计算

时间复杂度为$O(N^2)$

最大异或值

在给定的N个整数A1，A2……An中选出两个进行XOR运算，得到的结果最大是多少?

$N<=10^5$, $0<=Ai<2^{31}$

方法二:

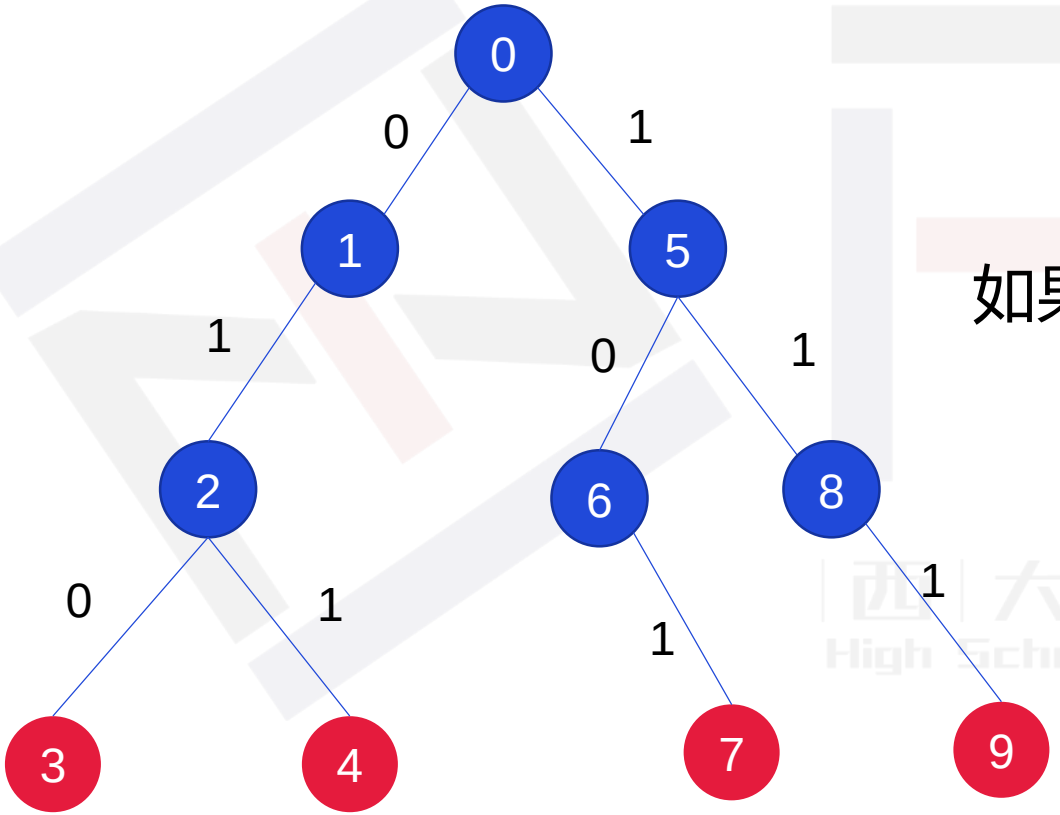
A和B进行XOR运算，将A、B转换为二进制后，每一位进行XOR。

N个正数A_i全部转换为32位二进制后建立字典树。

如何在字典树中找到XOR最大的?

最大异或值

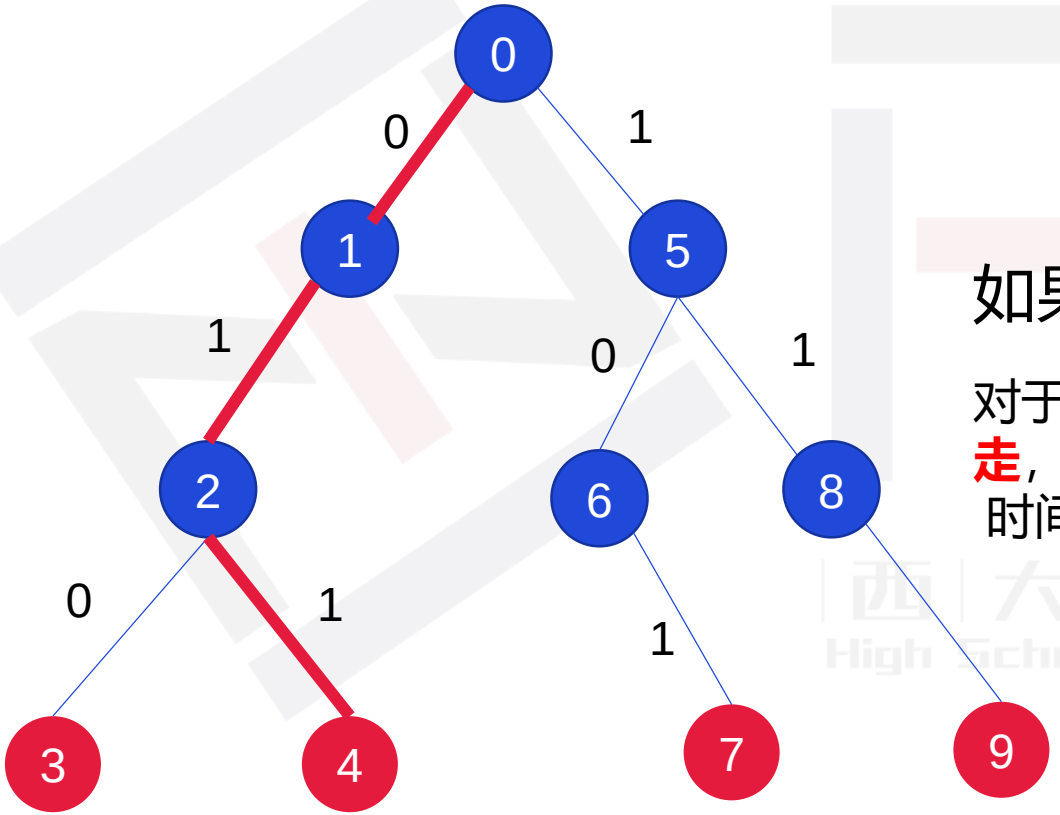
这里以3位二进制为例： 现在已经插入010(2)，011(3)，101(5)，111(7)。



如果现在要找与6(110)xor最大的，怎么找?

最大异或值

这里以3位二进制为例： 现在已经插入010(2)，011(3)，101(5)，111(7)。



如果现在要找与6(110)xor最大的，怎么找?

对于A_i每一位在字典树进行查找时，尽可能往与**该位相反的走**，最后得到的结果一定最大。

时间复杂度：O(32N)

字典序第k小

给定整数 n 和 k，找到 1 到 n 中字典序第 k 小的数字。

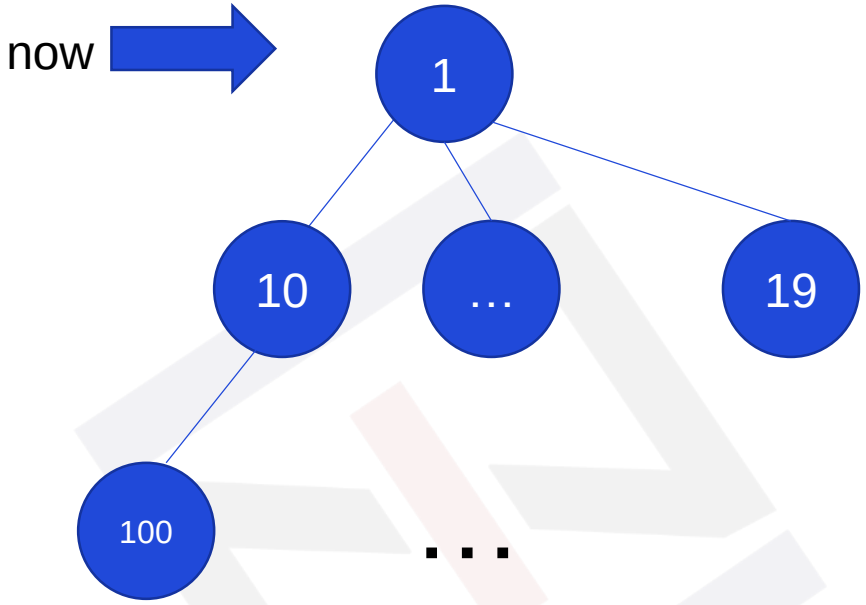
输入:
n: 13　k: 2
输出:
10

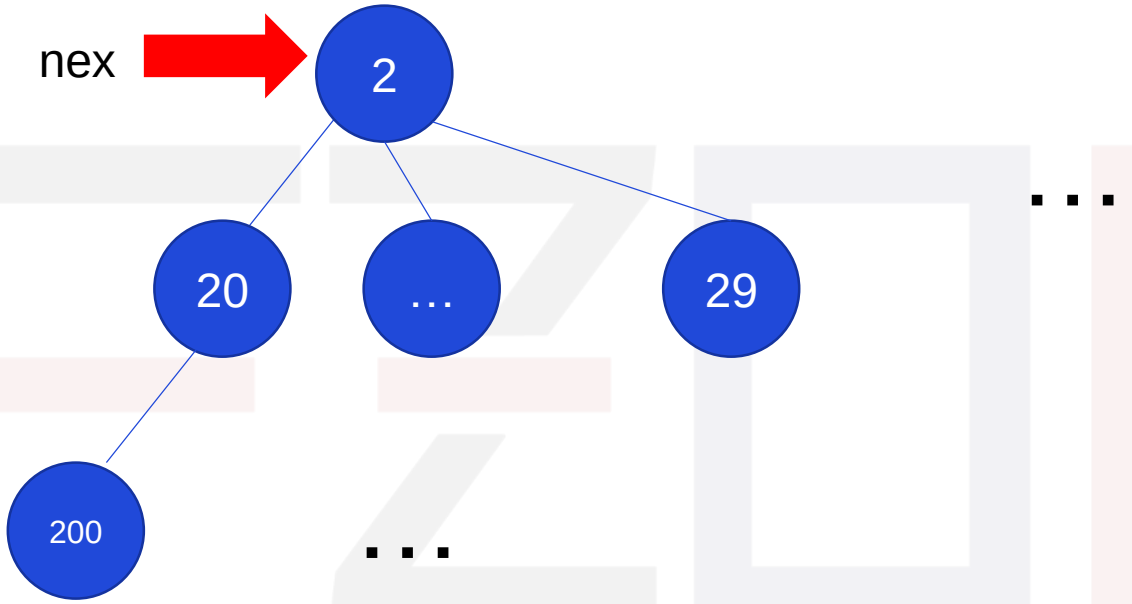
字典序的排列是 [1, 10, 11, 12, 13, 2, 3, 4, 5, 6, 7, 8, 9]，所以第二小的数字是 10。

1 <= k <= n <= 10^9

初步想法:
排序后输出
明显超时

字典序第k小





[1, 10, 11, 12, 13, 2, 3, 4, 5, 6, 7, 8, 9]

K=3，如何求?
K=6，如何求?

题目的本质就是在类似字典树结构中查找第k大的位置

先序遍历第k个，就是第k大的字典序

第一步是弄清楚树的结构
树的第一层默认有9个节点 分别是 1 – 9
同时默认每个节点默认有10个孩子，孩子节点也存在10个孩子

解析思路就是, 通过不断遍历来缩小范围

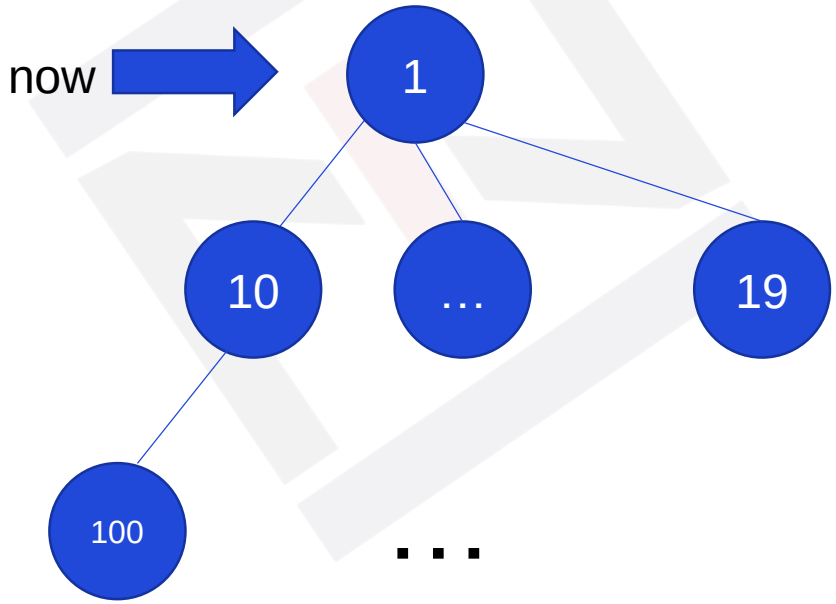
字典序第k小

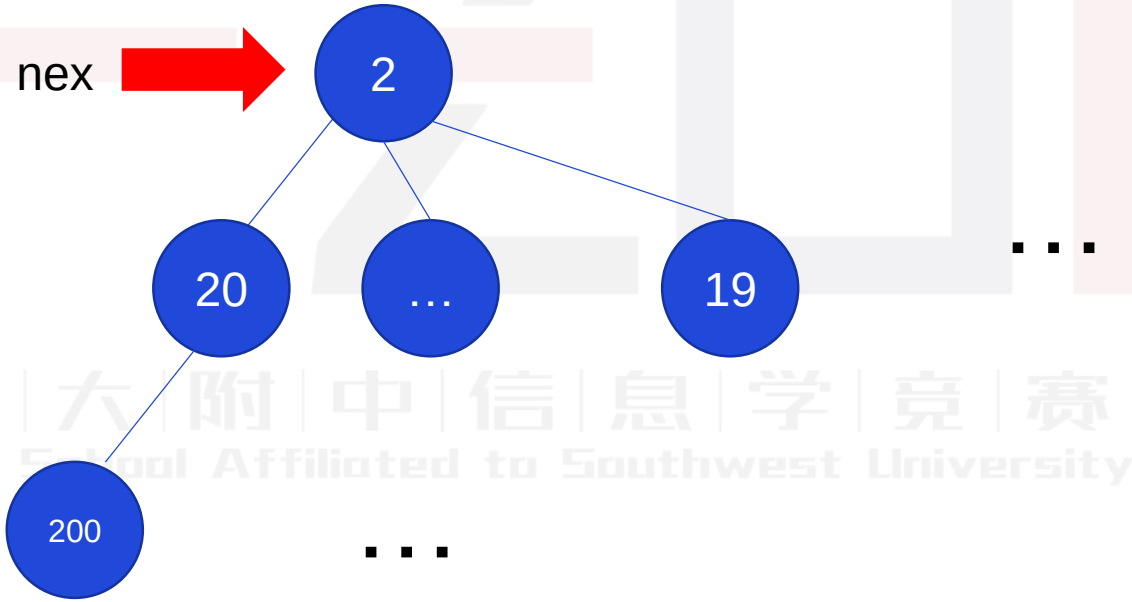
当前所在的节点now

往右走：
now=nex

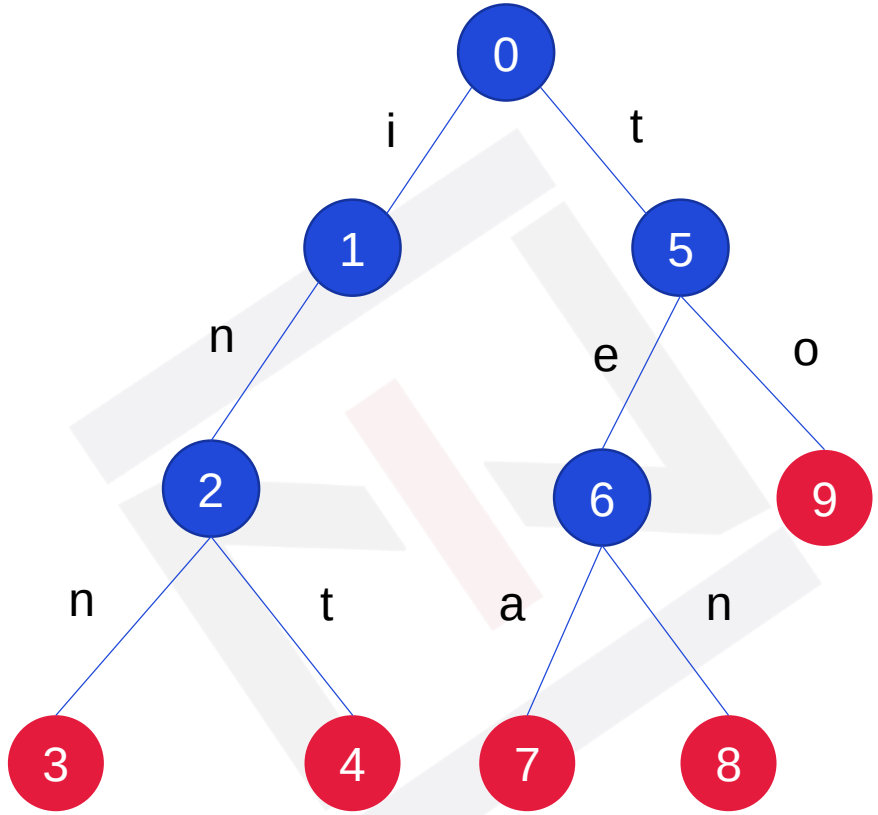
往下走：
now*=10

实现时需要考虑的细节：
- n与节点个数的关系





Trie树的优缺点



优点：
对于大部分字符串具有相同前缀的情况下，字典树能有效的节约空间

缺点：int Trie[MAX_NODE][CHARSET];
对于字母的trie来说CHARSET=26

但如果涉及到的字符过多，节点数也很多的时候，trie树的空间就炸了

应用

典型应用：词频统计、求前缀

优点：最大限度地减少无谓的字符串比较

核心思想：空间换时间。

利用字符串的公共前缀来降低查询时间的开销以达到提高效率的目的。

与hash的区别：trie支持O(logn)动态查询，hash只能离线查询

拓展了解

位运算trie

双数组版trie

后缀树

AC自动机(简化来说是Trie上进行KMP算法)

参考资源:

https://blog.csdn.net/weixin_45951804/article/details/104363998?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.essearch_pc_relevant&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7Edefault-1.essearch_pc_relevant

Thanks

For Your Watching

FZOI
西大附中信息学竞赛
High School Affilited to Southwest University