ta có 1 đỉnh là gốc và không lưu kí tự gì cả.

bây h ta cần xây dựng một cây trie cho các từ sau

ana

n

na

nna

oa

ona

thì có cái cây trên

ta sẽ đánh dấu tại các vị trí có dấu x để xác định đó là vị trí kết thúc một từ

thì khi đi duyệt chiều sâu dọc từ root tới 1 nút có dấu x, ta sẽ được 1 từ

vậy bây giờ xây dựng một nút của cây trie, t sẽ có 1 struct

struct Tnode{

int child[30]; /\*mảng này lưu 0,1 với ý nghĩ

child[i]=bằng 1 nếu có một từ có nút này mà có kí tự tiếp theo là i

=0 ngược lại

ví dụ ở cây trie trên, root có

child[‘a’-’a’],child[‘n’-’a’],child[‘o’-’a’]=1, còn lại bằng 0 hết

\*/

int pre; /\*là lưu nút cha của nút này. thứ tự nút cha,

không phải pre của n=a, mà pre của nút thứ i bằng nút thứ j, từ từ tí tí đề cập

\*/

int end; /\*nó là kết thúc của bao nhiêu từ\*/

int cnt; //thêm cái này là cái mà từ root tới nút này là xâu tiền tố của bao nhiêu từ

/\* tùy thuộc vào bài toán mà đôi khi còn cần nhiều thứ lưu trữ hơn, ví dụ từ root tới nút này là xâu tiền tố của bao nhiêu từ, hoặc với một số biểu diễn dưới dạng nhị phân thì cần biết tới đó là số bao nhiêu, rồi dưới nó có bao nhiêu nút con để quy về cây, xoay xiếc các kiểu,... ghê lắm chị k có học\*/

};

rồi bây h cái struct kia nó dùng như thế nào?

ta sẽ có 1 mảng

Tnode Trie[maxn]; //mảng lưu các nút của cây

int nT; //số lượng nút

int root; //gốc

bây h nút thứ 1 là root

mỗi khi thêm 1 kí tự thì ta sẽ có thêm 1 nút.

h nói đến cái pre

Trie[4].pre=3;

Trie[5].pre=1;

pre nó là lưu cái đó

hmm, nghĩ kĩ thì méo có thê làm quá 10^7 nút dc, làm cái mảng to đùng dùng là đúng :v

ta có hàm addword là thêm 1 từ

void addword(char \*s) //nói chung cái \* kia chị cũng k rõ lắm. thầy bảo code vậy thì code thôi

{

int r=root;

for(int i=0; s[i]!=0; ++i)

{

int k=s[i]-’a’;

int l;

if(!Trie[r].child[k]){

l=addnode();

Trie[r].child[k]=l;

Trie[l].pre=r;

}

r=Trie[r].child[k];

}

Trie[r].end++;

Trie[r].cnt++;

while(r!=root)

{

r=Trie[r].pre;

Trie[r].cnt++;

}

}

hàm thêm 1 kí tự.

int addnode()

{

++nT;

for(int i=0; i<26; ++i) Trie[nT].child[i]=0;

Trie[nT].pre=Trie[nT].end=Trie[nT].cnt=0;

return nT;

}

đếm xem bao nhiêu từ bằng s

int count(char \*s)

{

int r=root;

for(int i=0; s[i]!=0; ++i)

{

int k=s[i]-’a’;

if(trie[r].child[k]==0) return 0;

r=trie[r].child[k];

}

return Trie[r].end;

}

<http://vn.spoj.com/problems/CHAIN2/>

<http://vn.spoj.com/problems/SEC/>

<http://vn.spoj.com/problems/LTPMSEQ/>

<http://vn.spoj.com/problems/VOXOR/>

<http://codeforces.com/contest/706/problem/D>