Search Engine (Google)

Indexing.

KeyWord	Webpages
facebook	www.facebook

Web Crawler.

URL's

✓ URL1
✓ URL2
✓ URL3
URL4 ← URL1 URL3
⋮
URL10

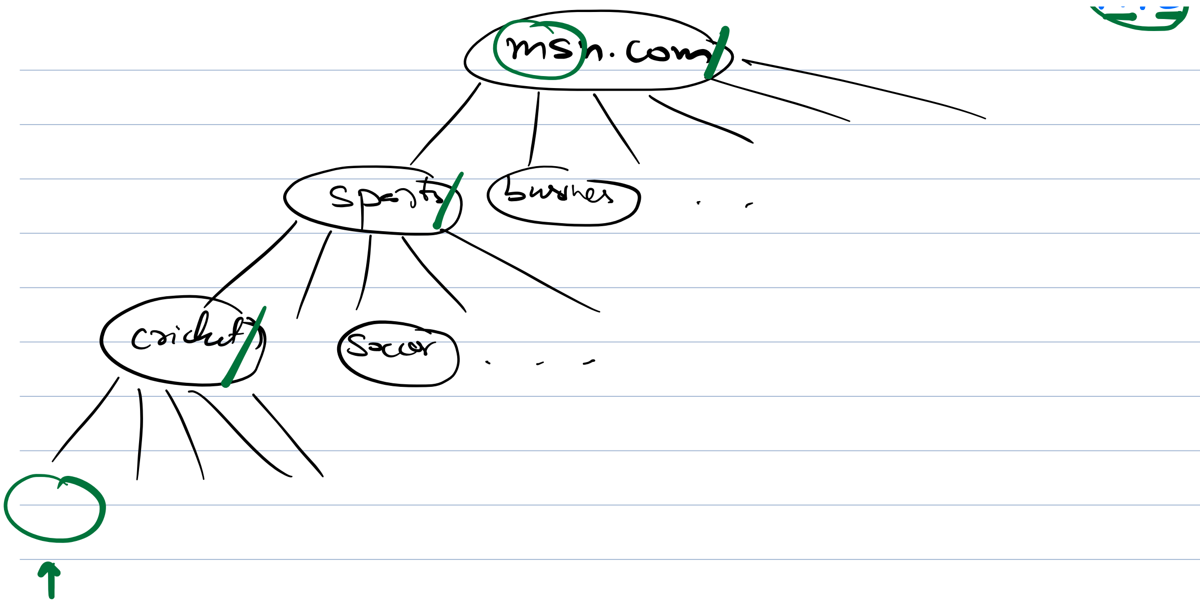
⇓

✓ Set of visited pages. Set (1 million)

msn.com / politics (100,000) ← cricket (10,000), football, tennis
sports
finance
economics
tourism
health

Tries (Tree like structure)

msn.com



RETRIEVAL ⇒ S.C.

⇒ Reduce space complexity by having shared space to store common prefix & also make retrieval easier.

Prashant
Utkarsh
✓ Ankit

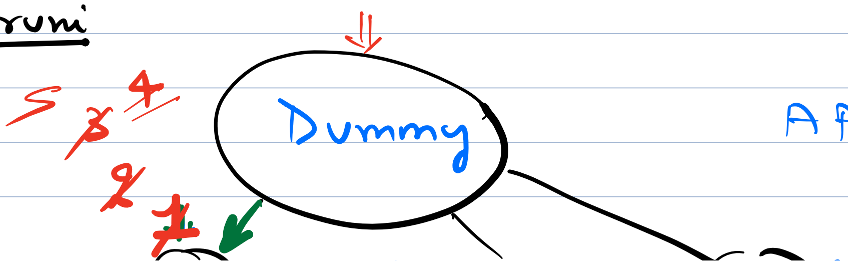
Varun ✓
Tarun ✓
Taruni

Chitra
Anurag
Vishal
Vimal

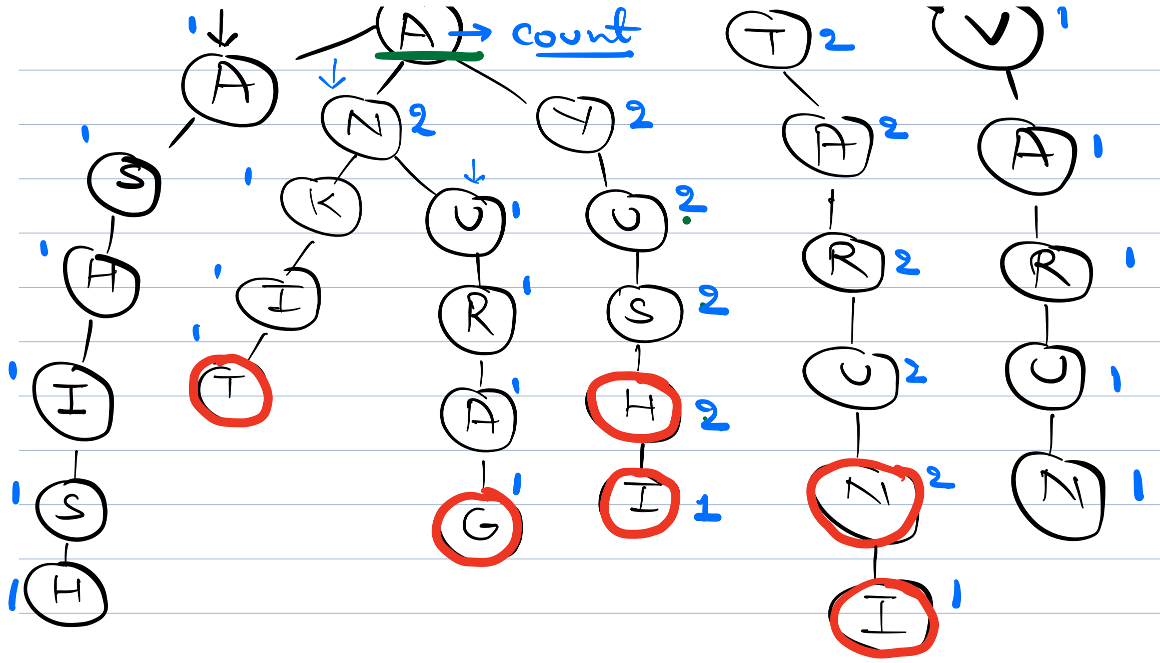
Ayush

Ayushi ✓

KRISH



AASHISH



Space = Cap Letters

↓

26 children

[A - Z]

A — 0
B — 1
⋮
Z — 25

```
Node {

        char   c;

        Node     children[26];  →
        boolean    isEnd;
}

        HashMap<char, Node>
                      children
```

1) Insert

```
void   insert ( root, word) {
        Node   curr =  root ;

        for ( i=0;  i< word. length();  i++) {
                ch =  word. charAt(i);
                if(! curr. children. containsKey(ch)) {
                        curr.children. put (
                         ch,  new  Node (ch));
                }
               curr =  curr. children.get(ch);
        }

        Curr.  isEnd =  true;

}
```

T.C. = O(Length)

```
boolean  search  ( root,   word) {
        Node   curr  =   root;

        for (i=0;  i<  word. length();  i++) {
                ch =  word.  charAt (i);
        if (! curr.children.containsKey(ch)){
            return false;
        }
        curr = curr.children.get(ch);
    }
    return curr.isEnd;
}
```

T.C. = O(Length of word)

String ⇒ Hash

H.W: Delete a Word

Q Given an array of words.(strings) Return an array of strings containing the smallest unique prefix for every word.

(Ass: Unique prefix will exist for every word)

A: [cat, dog, rat, tiger, racoon]

↓ ↓ ↓ ↓ ↓

c d rat t rac

A: [dog, doog, doc, donkey, duck, done]

↓ ↓ ↓ ↓ ↓ ↓

dog doo doc donk du done

H.W.

1) Insert

2) Search