Tries
Tries

• We have seen a few data structures that handle the mapping of key-value pairs.
  • Arrays: The key is the element index, the value is the data at that location.
  • Hash tables: The key is the hash code of the data, the value is a linked list of data hashing to that hash code.

• What about a slightly different kind of data structure where the key is guaranteed to be unique, and the value could be as simple as a Boolean that tells you whether the data exists in the structure?
Tries

• Tries combine structures and pointers together to store data in an interesting way.

• The data to be searched for in the trie is now a roadmap.
  • If you can follow the map from beginning to end, the data exists in the trie.
  • If you can’t, it doesn’t.

• Unlike with a hash table, there are no collisions, and no two pieces of data (unless they are identical) have the same path.
Tries

• Let’s map key-value pairs where the keys are four-digit years (YYYY) and the values are names of universities founded during those years.

• In a trie, the paths from a central root node to a leaf node (where the school names would be), would be labeled with digits of the year.

• Each node on the path from root to leaf could have 10 pointers emanating from it, one for each digit.
Tries

• To insert an element into the trie, simply build the correct path from the root to the leaf.
Tries

typedef struct _trie
{
    char university[20];
    struct _trie* paths[10];
}
trie;
typedef struct _trie {
    char university[20];
    struct _trie* paths[10];
} trie;
Tries
Tries

Insert “Harvard”, founded 1636
Tries

Insert “Harvard”, founded 1636
Tries

Insert “Harvard”, founded 1636
Tries

Insert “Harvard”, founded 1636
Tries

Insert “Harvard”, founded 1636
Tries

Insert “Harvard”, founded 1636
Insert “Harvard”, founded 1636
Tries

Insert “Harvard”, founded 1636
Tries

Insert “Harvard”, founded 1636
Tries

```
Harvard
```
Tries

Insert “Yale”, founded 1701

“Harvard”
Tries

Insert “Yale”, founded 1701

“Harvard”
Insert “Yale”, founded 1701
Tries

Insert “Yale”, founded 1701
Tries

Insert “Yale”, founded 1701

“Harvard”
Tries

Insert “Yale”, founded 1701

“Harvard”
Tries

Insert “Yale”, founded 1701
Tries

```
Harvard
```

```
Yale
```
Insert “Dartmouth”, founded 1769

Tries

“Harvard”

“Yale”
Insert “Dartmouth”, founded 1769
Insert “Dartmouth”, founded 1769
Insert “Dartmouth”, founded 1769

Tries

“Harvard”

“Yale”
Insert “Dartmouth”, founded 1769

Tries

“Harvard”

“Yale”
Insert “Dartmouth”, founded 1769

Tries

“Harvard”

“Yale”
Tries

Insert “Dartmouth”, founded 1769
Tries

- "Harvard"
- "Yale"
- "Dartmouth"
Tries

• To search for an element in the trie, use successive digits to navigate from the root, and if you can make it to the end without hitting a dead end (a NULL pointer), you’ve found it.
Search for “Harvard”, founded 1636
Tries

Search for “Harvard”, founded 1636

“Harvard”

“Yale”

“Dartmouth”
Search for “Harvard”, founded 1636
Search for “Harvard”, founded 1636

“Harvard”

“Yale”

“Dartmouth”
Search for “Harvard”, founded 1636

Tries

“Harvard”

“Yale”

“Dartmouth”
Tries

Search for “Harvard”, founded 1636

“Harvard”

“Yale”

“Dartmouth”
Tries

Search for “Harvard”, founded 1636
Search for “Princeton”, founded 1746

“Harvard”

“Yale”

“Dartmouth”
Search for “Princeton”, founded 1746
Tries

Search for “Princeton”, founded 1746

“Harvard”

“Yale”

“Dartmouth”
Tries

Search for “Princeton”, founded 1746

“Harvard”

“Yale”

“Dartmouth”
Search for “Princeton”, founded 1746
Tries

Search for “Princeton”, founded 1746

“Harvard”

“Yale”

“Dartmouth”