Tries

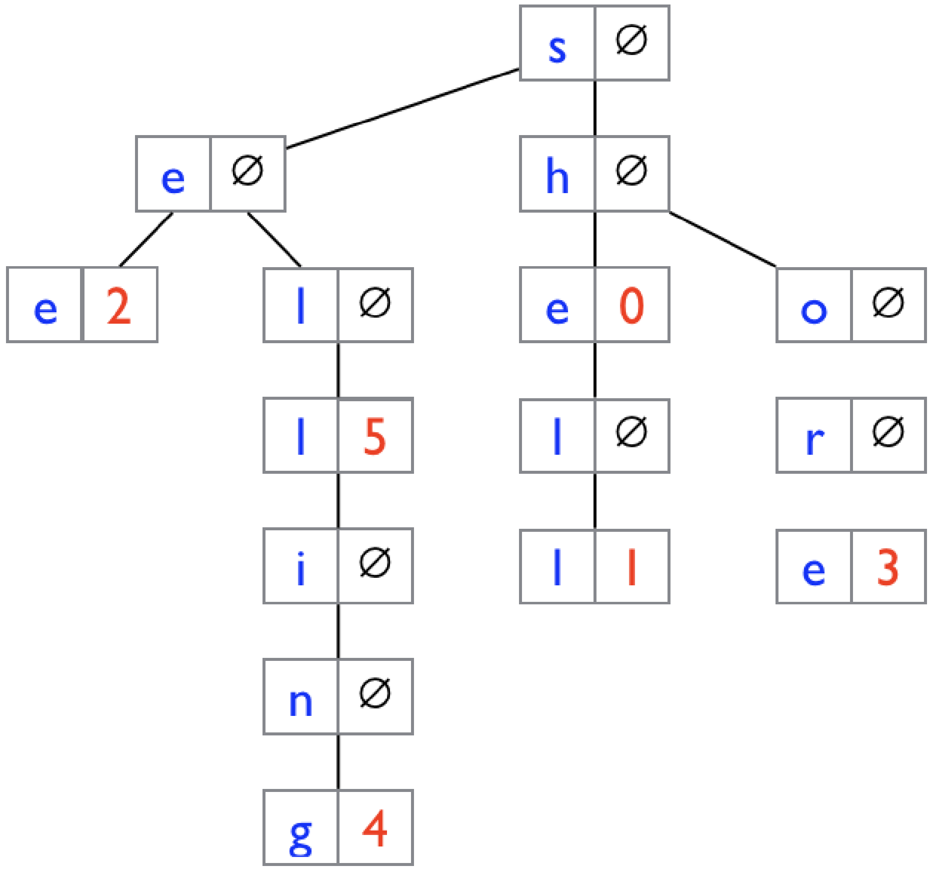
Contents

Trie vs. BST

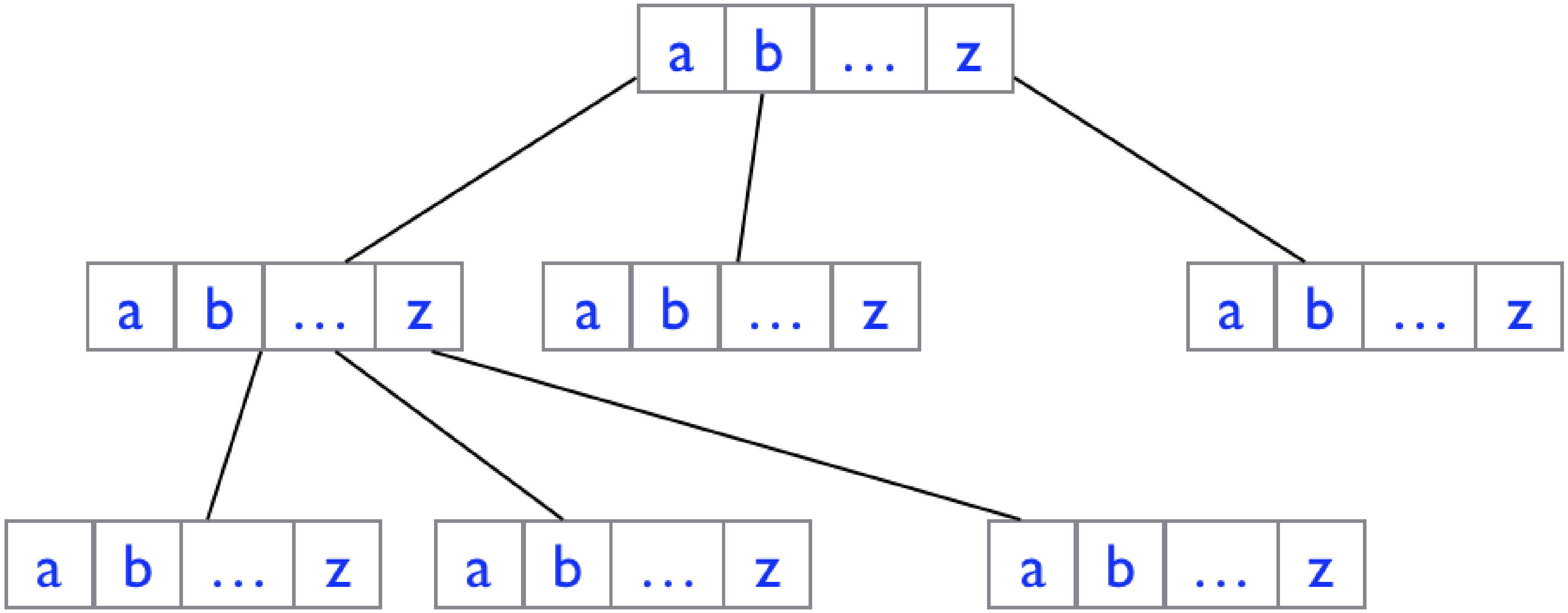
String comparisons

```
{"she", "shell", "see", "shore", "selling", "sell"}
```

- How many character comparisons using a binary search tree or a trie?



Children representation



- How many childrens are expected per node?
- What data strucctures to represent childnre?

Trie Node

```
public class TrieNode<T> {
    private Map<Character, TrieNode<T>> children;
    private TrieNode<T> parent;
    private boolean end_state;
    private char key;
    private T value;

    public TrieNode(TrieNode<T> parent, char key) {
        children = new HashMap<Character, TrieNode<T>>();
        setEndState(false);
        setParent(parent);
        setKey(key);
        setValue(null);
    }
```

- Generic: what is `<T>` for?
- Base API: HashMap.
- Member instances: `value` vs. `end_state`.

Source: `TrieNode.java`

```java
public TrieNode<T> getChild(char key) {
    return children.get(key);
}

public TrieNode<T> addChild(char key) {
    TrieNode<T> child = getChild(key);

    if (child == null) {
        child = new TrieNode<T>(this, key);
        children.put(key, child);
    }

    return child;
}

public TrieNode<T> removeChild(char key) {
    return children.remove(key);
}
```

- Complexities: `get()`, `add()`, `remove()`.

Trie

Source: `Trie.java`

```java
public class Trie<T> {
    private TrieNode<T> root;

    public Trie() {
        root = new TrieNode<>(null, (char) 0);
    }

    public T get(String key) {
        TrieNode<T> node = find(key);
        return (node != null && node.isEndState()) ? node.getValue() : null;
    }

    public TrieNode<T> find(String key) {
        char[] array = key.toCharArray();
        TrieNode<T> node = root;

        for (int i = 0; i < key.length(); i++) {
            node = node.getChild(array[i]);
            if (node == null) return null;
        }

        return node;
    }
```

- Dummay character: `(char)0`.
- Complexity: `find()`.

```java
public T put(String key, T value) {
    char[] array = key.toCharArray();
    TrieNode<T> node = root;

    for (int i = 0; i < key.length(); i++)
        node = node.addChild(array[i]);

    node.setEndState(true);
    return node.setValue(value);
}
```

```java
public boolean remove(String key) {
    TrieNode<T> node = find(key);

    if (node == null || !node.isEndState())    // node doesn't exist
        return false;

    if (node.hasChildren()) {    // node to be removed as children
        node.setEndState(false);
        return true;
    }

    TrieNode<T> parent = node.getParent();

    while (parent != null) {    // remove iteratively
        parent.removeChild(node.getKey());

        if (parent.hasChildren() || parent.isEndState())   // another word
            break;
        else {
            node = parent;
            parent = parent.getParent();
        }
    }

    return true;
}
```

References

- Trie.