Week 13 - Friday

COMP 2100

Last time

- What did we talk about last time?
- Heap implementation
- Heap sort

Questions?

Project 4

Assignment 7

Finish Bubble Down

Timsort

Timsort

- Timsort is a recently developed sorting algorithm used as the default sort in Python
- It is also used to sort non-primitive arrays in Java
- It's a hybrid sort, combining elements of merge sort and insertion sort
- Features
  - Worst case and average case running time: O(***n*** log ***n***)
  - Best case running time: O(***n***)
  - Stable
  - Adaptive
  - Not in-place

Ideas behind Timsort

- We also want to find "runs" of data of two kinds:
  - Non-decreasing: 34, 45, 58, 58, 91
  - Strictly decreasing: 85, 67, 24, 18, 7
- These runs are already sorted (or only need a reversal)
- If runs are not as long as a minimum run length determined by the algorithm, the next few values are added in and sorted
- Finally, the sorted runs are merged together
- The algorithm can use a specially tuned galloping mode when merging from two lists
  - Essentially copying in bulk from one list when it knows that it won't need something from the other for a while

Timsort implementation

- It might be useful to implement Timsort in class, but it has a lot of special cases
- It was developed from both a theoretical perspective but also with a lot of testing
- If you want to know more, read here:
  - https://www.infopulse.com/blog/timsort-sorting-algorithm/

Sort visualizations

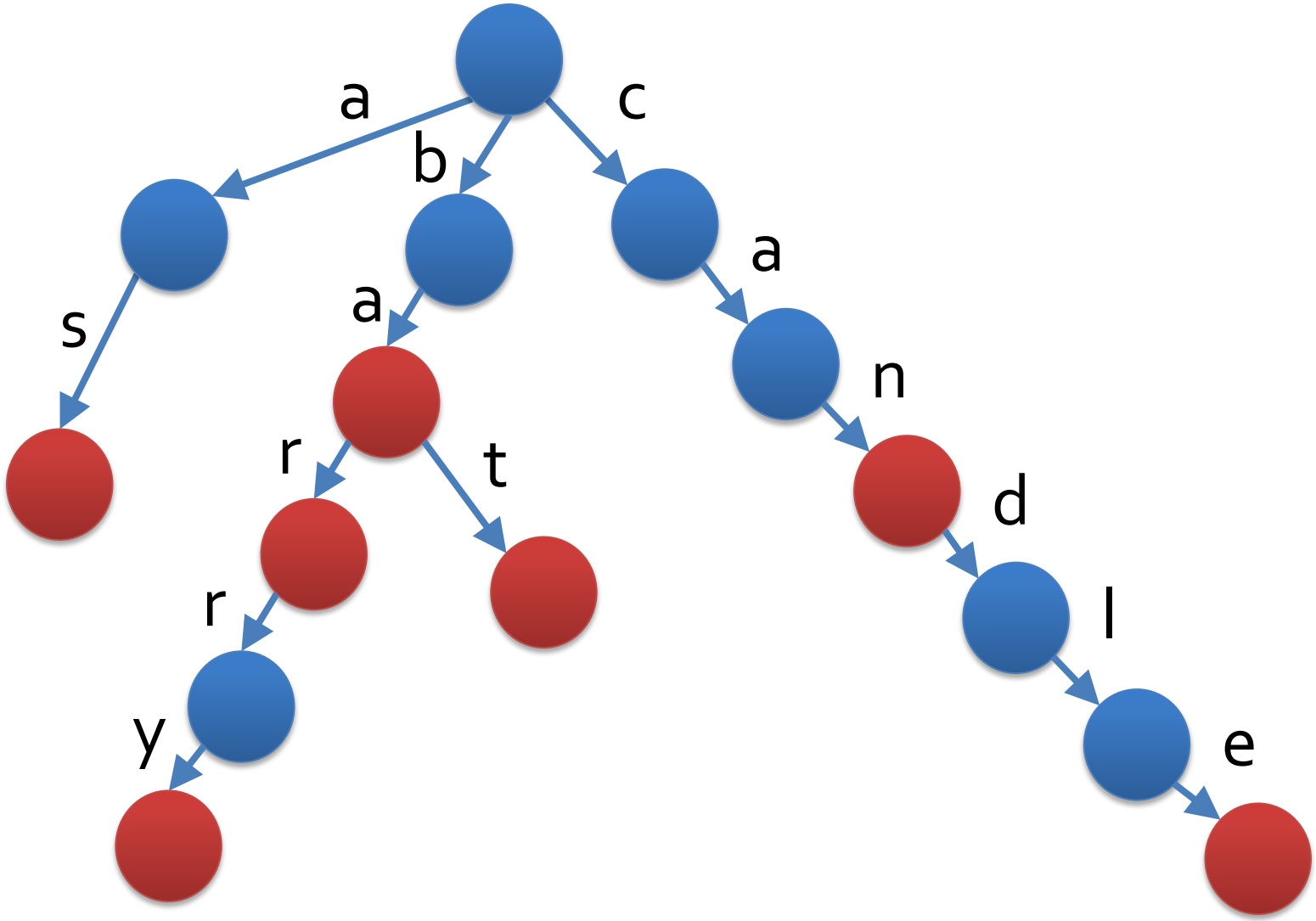
- Understanding how sorts work can be challenging
- Understanding how running time is affected by various algorithms and data sets is not obvious
- To help, there are many good visualizations of sorting algorithms in action:
  - http://www.youtube.com/watch?v=kPRAoW1kECg
  - https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

Tries

Storing strings (of anything)

- We can use a (non-binary) tree to record strings implicitly where each link corresponds to the next letter in the string
- Let's store:
  - ba
  - bar
  - bat
  - barry
  - can
  - candle
  - as

Trie this on for size



Trie practice

- Now you add:
  - he
  - she
  - her
  - help
  - sat
  - rat

Trie implementation

```
public class Trie {
  private static class Node {
    public boolean terminal = false;
    public Node[] children = new Node[128];
  }

  private Node root = new Node();
}
```

Trie Contains

Signature for recursive method:

```
private static boolean contains(Node node, String
  word, int index)
```

Called by public proxy method:

```
public boolean contains(String word) {
  return contains(root, word, 0);
}
```

Trie Insert

Signature for recursive method:

```
private static void insert(Node node, String word,
  int index)
```

Called by public proxy method:

```
public void insert(String word) {
  insert(root, word, 0);
}
```

Trie Traversal

```
private static void inorder(Node node, String prefix)
```

Called by public proxy method:

```
public void inorder(String word) {
  inorder(root, "");
}
```

Cost

- Let ***m*** be the length of a particular string
- Find Costs:
  - O(***m***)
- Insert Costs:
  - O(***m***)

Trie implementations

- Keeping an array of length equal to all possible characters (usually) wastes space
- Alternatives:
  - **Ternary search tries:** A lot like a binary search tree, with smaller characters to the left, larger characters to the right, and continuations from the current character beneath
  - Keeping an array (or linked list) of the characters used, resizing as needed

Quiz

Upcoming

Next time…

- Finish tries
- Substring search

Reminders

- **1:45 – 2:45 office hours canceled today because of AI Task Force**
- Work on Project 4
- Finish Assignment 7
  - **Due tonight by midnight!**
- Read Section 5.3