Trie Adventure

Node Definitions For A Smaller Footprint

Originally Developed by
Keith Hellman

Heavily Modified by
Anastasia Sokol

Updated on June 2, 2024

Introduction

We have studied the abstract trie data structure, a specialized tree designed for the quick search of sequenced data such as words, and we also implemented a trie in 3P. Both the lecture and 3P version store a pointer to each of the 26 possible subsequent lowercase letters in *each node*, which is *208 bytes* on most machines, and means that storing the word ”algorithms” takes almost two kilobytes!

In this project you will explore some specialized node structures to decrease the memory footprint of the trie structure. As you go, write responses to the bolded, numbered questions along with any properly formatted graphs, charts, or equations that the question asks for. This project has a longer prompt that some of the past assignments, but most of the content is there to guide you through the project, and the deliverables are equivalent to – if not less than – other analysis projects in this class.

tl;dr This project is about decreasing the memory footprint of the trie class. Answer the bold questions. *It is not as much work as it seems.* If you get stuck or have questions do not hesitate to reach out for help after class or in office hours. Good luck!

The Default Trie

A Bitmasked Trie

Empirical Measurements

Rubric

Language Feature Explanations