Computer Science 222: Succincter

Tuesday, December 11, 2012

Professor Mitzenmacher

Dan Bradley (dbradley@college), Saagar Deshpande (sdeshpande@college)

Final Project

****

1 Abstract

Using Mihai Patrascu's 2008 paper "Succincter", we implement a way to store trits (trinary values) within 1.05% of the ideal space of $n * log_2(3)$ while having lookup in $O(t)$ time, where t is the depth of our data structure. We find that this is both a fast and space efficient data structure with room for extension past simply storing trits.

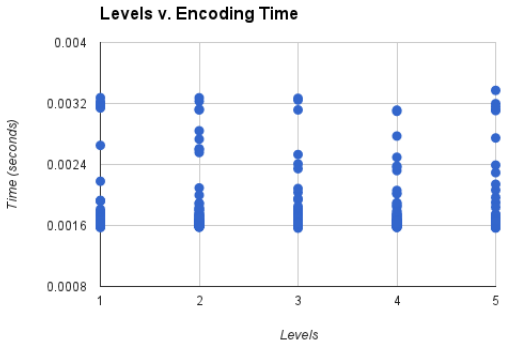
2 Introduction

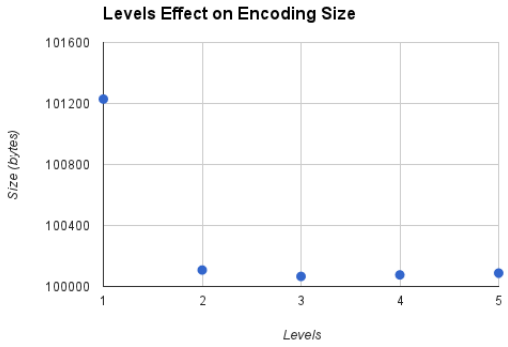
There are few effective methods for storing trits. In this paper, we focus on three of them: the naive method, arithmetic coding, and our succincter implementation.
The naive method is simple. As a trit has more information than a bit, but less than two bits, store each trit using two bits. Obviously this is not the most space-efficient method, you could encode $\frac{4}{3}$ as much information in the same space, so it is clearly wasteful. It is however, very fast, both on encode and decode. Encode is strictly linear, with very small constants. The encoder we used took 3.9 milliseconds to encode 500000 entries, and 13.7 milliseconds to decode 500000 entries.
Arithmetic coding is much more effective at reducing size. By reducing the entries to one number, we can encode in exactly $n * log_2(3) + c$, where c is the

3 Implementation

4 Results and Analysis

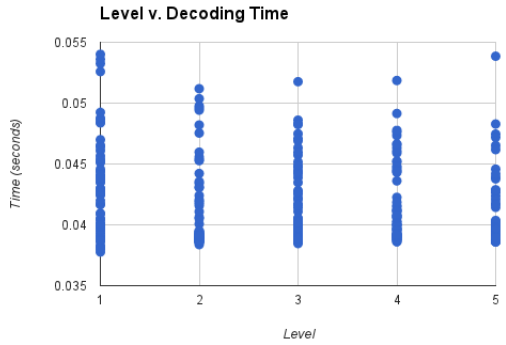
We ran our implementation of Succincter on 500000 trits except in the case of individual decoding, where we only use 200000 trits. We compared this implementation against a simple C implementation of arithmetic encoding as well as a naive C implementation of a trit to bit converter, which stores each trit as its corresponding bit.

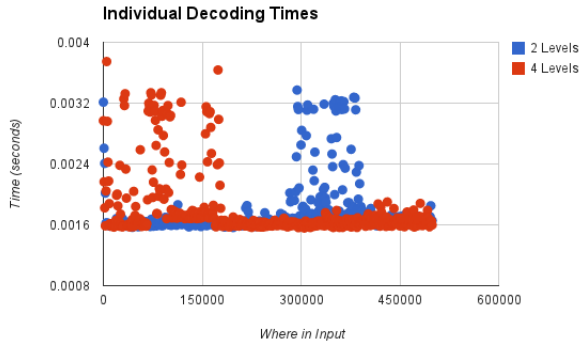




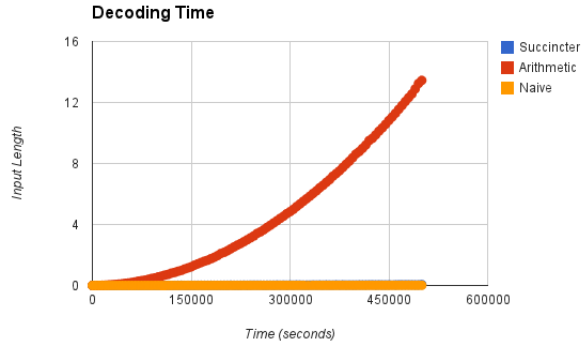
Initially, we compared the speed of encoding and decoding at each level.

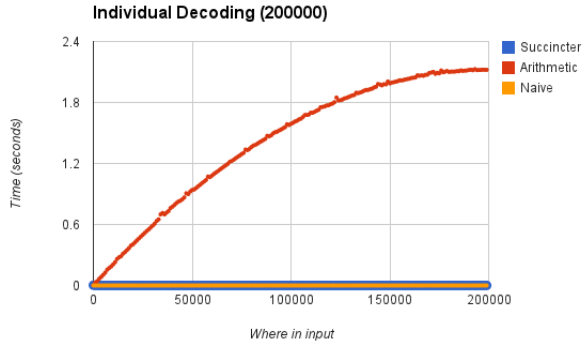
Level	Avg. Encode Time	Avg. Decode Time	Avg. Encode Size (bytes)
1	0.00178906	0.043107825396825	101229
2	0.00177046	0.041240047619048	100108
3	0.00173798	0.041672920634921	100066
4	0.00173344	0.041674126984127	100076
5	0.00175082	0.040755396825397	100088



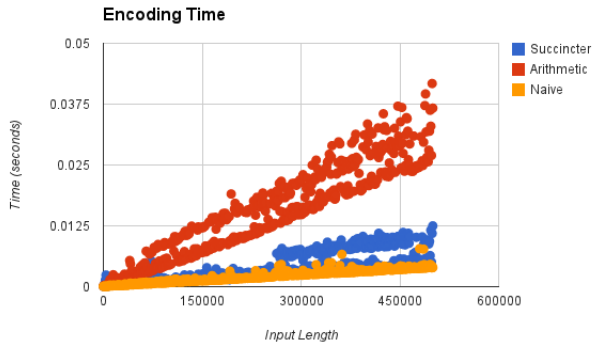


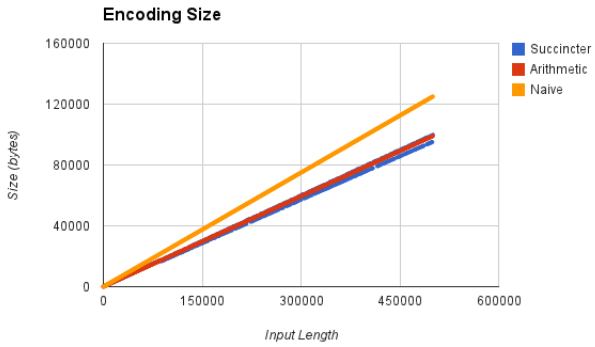
asdfaksdjf;aksldfjadsf
asdfaksdjf;aksldfjadsf
asdfaksdjf;aksldfjadsf





asdfaksdjf;aksldfjadsf
asdfaksdjf;aksldfjadsf
asdfaksdjf;aksldfjadsf
asdfaksdjf;aksldfjadsf





asdfaksdjf;aksldfjadsf

5 Conclusion

****

Appendix