CS166 Project Proposal (in Ranked Order)

Aaron Effron, Magdy Saleh, Hassan Fahmy

April 2019

1 Bloomier Filters

Having been exposed to bloom filters before we are really excited to further explore how layering them together allows for map encoding. We are particularly interested in understanding more about their construction and the mentioned neural network weight compression application.

Our initial sources are the seminal Chazelle paper introducing Bloomier filters and the build up paper by Charles that gives an improved construction algorithm.

1.1 Papers

- Bloom filters for arbitrary functions: `https://www.cs.princeton.edu/~chazelle/pubs/soda-rev04.pdf`
- Build Bloomier filters faster than original paper: `https://arxiv.org/pdf/0807.0928.pdf`

2 Approximate Distance Oracles

As well-dressed graph theory enthusiasts, we are interested in all shortest-path problems. Most of the work we've done in the past has been on exact algorithms, and here we are very interested in exploring a more real-world application in which we may slightly sacrifice complete accuracy for efficiency.

Our first initial source is the original paper presenting Approximate Distance Oracles by Thorup and Zwick, which involve $O(kmn^{1/k})$ expected time preprocessing and $O(k)$ approximate query times.

Our second is a followup to the original paper, by Chechik, which provides a distance oracle of size $O(kn^{1+1/k})$ with constant query time. There are multiple papers referenced by Chechik's work that we would also plan to use in our project (e.g., Wulff Nielsen's work on Approximate Distance Oracles with improved preprocessing time (`https://arxiv.org/pdf/1109.4156.pdf`) and improved query time (`https://arxiv.org/pdf/1202.2336.pdf`)

2.1 Papers

- Orignal paper presenting ADO's: https://www.cs.bgu.ac.il/~elkinm/thorup_zwick.do.pdf
- Chechik, Constant query time: https://arxiv.org/pdf/1305.3314.pdf
- Wulff Nielsen, improved preprocessing https://arxiv.org/pdf/1109.4156.pdf
- Wulff-Nielsen, improved query time https://arxiv.org/pdf/1202.2336.pdf

3 Decremental Tree Connectivity

Another very interesting problem relating to graph theory. We are interested in understanding how algorithms change when the problem concerns dynamic graphs. Specifically, we are really excited that this problem covers both graph aspects as well as machine word tricks similar to Fischer-Heun RMQs. As our sources we are considering the initial paper by Alstrup and then slides from TUD that break down the paper and its methods. Finally, we look at a paper by Thorup that presents a randomized alternative.

3.1 Papers

- $O(n + m)$ algorithm for maintaining tree connectivity on m deletions https://tinyurl.com/y3a3z2wd
- Slides from Phillip Bille from the Technical University of Denmark presenting the problem of Decremental Tree Connectivity: https://tinyurl.com/yxjrmay2
- Improved runtimes for maintaining spanning forest for m deletions https://tinyurl.com/yxryz6tx

4 Suffix Trees for Matrix Multiplication

We were already amazed by the idea of using suffix trees for substring search in class, and were convinced that they are an awesome data structure. The fact that they can also be used for matrix multiplication, obviously a hugely useful task across Computer Science, makes us even more excited to dive into Suffix Trees.

Our first source uses suffix trees to speed up matrix multiplication in n-gram based algorithms http://theory.stanford.edu/~hpaskov/pub/cr_uai.pdf. Our second source uses suffix trees to speed up string kernels, which compute all common substrings between two strings http://www.stat.purdue.edu/~vishy/papers/TeoVis06.pdf. Both papers explore the memory savings

attributed to using suffix trees in their various contexts, which we would love to explore.

4.1 Papers

- Speed up N-gram based ML algorithms (or just those that involve multiplication!) `http://theory.stanford.edu/~hpaskov/pub/cr_uai.pdf`
- Space efficient string kernels (compare all substrings between two strings) `http://www.stat.purdue.edu/~vishy/papers/TeoVis06.pdf`