Compressed data structures
Jeffrey Vitter

The world is drowning in data. The influx of massive data sets present a number of difficulties with storage and data accessibility. A primary computing challenge in all of these cases is how to compress the data but still allow it to be queried quickly. In this talk, we give a brief introduction towards the design of compressed data structures that operate in near-optimal time bounds, while using a near-minimum amount of space in terms of the particular instance of input data. We call such structures ``data-aware,'' since they exploit the compressibility of the particular input. To illustrate the paradigm, we showcase some of our ongoing research in text indexing and dictionaries.

For text indexing, we build upon the well-known suffix array, which is an important data structure for many indexing tasks. Given a text T of n characters, a suffix array is an array that specifies the positions of each of the n suffixes of T in lexicographical order. We describe a special type of compressed suffix array exhibiting new tradeoffs between search time and space occupancy. These entropy-compressed suffix arrays are nearly optimal in space, requiring exactly the empirical entropy of T plus negligible additional space, and they allow fast search queries. We achieve our entropy space bound by exploiting the strong overlap between consecutive suffixes of the text T to encode each suffix more succinctly.

As a direct consequence of our methods, we can also represent the Burrows-Wheeler transform using nearly optimal space. For dictionaries, we consider the operations of membership, predecessor, rank, and select. The data set consists of a subset S of n items from a universe U. We use a measure called gap to exploit the compressibility of particular data sets. We go on to describe a novel dictionary structure operating in nearly optimal space (according to the gap measure) and in time close to the lower bound for predecessor searching. Our primary building block describes a method to search a gap-encoded stream of numbers using a binary search rather than the usual linear scan. For real-life data, we experimentally show the value of a gap-style encoding over other methods that are not data-aware.