This lecture describes a tour-de-force in integer data structures, called fusion trees (so named because they were published a year after the 1989 cold-fusion "scandal", and were perhaps just as surprising—though more correct). Fusion trees solve predecessor and successor among \( n \) \( w \)-bit integers in \( O(\log_w n) \) time per operation on the word RAM. The basic idea is to build a B-tree with branching factor \( w^\varepsilon \). The tricky part is to build a \( w^\varepsilon \)-size node supporting constant-time predecessor/successor.

We'll see three major techniques for doing so. First, sketching reduces the \( w^{1+\varepsilon} \) bits in a node down to \( w \) “essential” bits, by reducing each word down to \( w^{1-\varepsilon} \) “essential” bits. The impressive part is sketching a word in constant time using a single multiplication. Second, parallel comparison lets us compare all of these sketches with a single query in constant time. Third, we'll see how to find the most significant set bit of a \( w \)-bit word in constant time on a word RAM, by using most of the fusion techniques again; this problem has tons of applications beyond fusion trees as well. (As a result, most significant set bit is a built-in instruction on most architectures; see e.g. StackOverflow.)